PHOTOGRAPH THIS SHEET

AD-A200 902

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

AFWAL-TR-88-1076

DOCUMENT IDENTIFICATION

Sept 1988

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS GRA&I	☒
DTIC TAB	☐
UNANNOUNCED	☐
JUSTIFICATION	

BY

DISTRIBUTION /
AVAILABILITY CODES

DIST	AVAIL AND/OR SPECIAL

A-1

DTIC
COPY
INSPECTED
6

DISTRIBUTION STAMP

DTIC
ELECTE
DEC 0 2 1988
E

DATE ACCESSIONED

DATE RETURNED

8 8 12 2 002

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NO.

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

DTIC FORM 70A
MAR 86

DOCUMENT PROCESSING SHEET

PREVIOUS EDITION MAY BE USED UNTIL
STOCK IS EXHAUSTED.

AFWAL-TR-88-1076

BIOMASSCOMP

AD-A200 902

Robert L. Dawes

Martingale Research Corporation
100 Allentown Parkway, Suite 211
Allen, TX   75002

September 1988

Final Report for Period August 1987 - February 1988

Approved for Public Release; Distribution is Unlimited

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
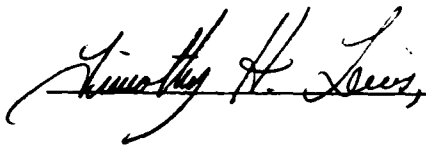WRIGHT-PATTERSON AIR FORCE BASE, OHIO   45433-6543

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
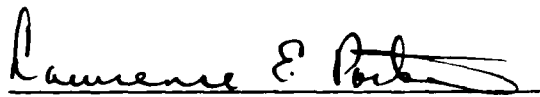NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

NOTICE

WHEN GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA ARE USED FOR ANY PURPOSE OTHER THAN IN CONNECTION WITH A DEFINITELY GOVERNMENT-RELATED PROCUREMENT, THE UNITED STATES GOVERNMENT INCURS NO RESPONSIBILITY OR ANY OBLIGATION WHATSOEVER. THE FACT THAT THE GOVERNMENT MAY HAVE FORMULATED OR IN ANY WAY SUPPLIED THE SAID DRAWINGS, SPECIFICATIONS, OR OTHER DATA, IS NOT TO BE REGARDED BY IMPLICATION, OR OTHERWISE IN ANY MANNER CONSTRUED, AS LICENSING THE HOLDER, OR ANY OTHER PERSON OR CORPORATION; OR AS CONVEYING ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY IN ANY WAY BE RELATED THERETO.

THIS REPORT HAS BEEN REVIEWED BY THE OFFICE OF PUBLIC AFFAIRS (ASD/CPA) AND IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

LAWRENCE E. PORTER
Chief, Plans Branch
Avionics Laboratory

FOR THE COMMANDER

JAMES W, FALTER, Chief
Plans and Operations Office
Avionics Laboratory

IF YOUR ADDRESS HAS CHANGED, IF YOU WISH TO BE REMOVED FROM OUR MAILING LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION PLEASE NOTIFY AFWAL/AAOR , WRIGHT-PATTERSON AFB, OH 45433- 6543 TO HELP US MAINTAIN A CURRENT MAILING LIST.

COPIES OF THIS REPORT SHOULD NOT BE RETURNED UNLESS RETURN IS REQUIRED BY SECURITY CONSIDERATIONS, CONTRACTUAL OBLIGATIONS, OR NOTICE ON A SPECIFIC DOCUMENT.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS			
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Distribution			
2b DECLASSIFICATION/DOWNGRADING SCHEDULE		Approved for Public Release; Distribution is Unlimited			
4 PERFORMING ORGANIZATION REPORT NUMBER(S) MRC-WPAFB-88-001		5 MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-88-1076			
6a NAME OF PERFORMING ORGANIZATION MARTINGALE RESEARCH CORPORATION	6b OFFICE SYMBOL (if applicable) ADVPHEN	7a. NAME OF MONITORING ORGANIZATION Avionics Laboratory (AFWAL/AAOR) Air Force Wright Aeronautical Laboratories			
6c. ADDRESS (City, State, and ZIP Code) 100 ALLENTOWN PARKWAY, SUITE 211 ALLEN, TX 75002		7b ADDRESS (City, State, and ZIP Code) WRIGHT-PATTERSON AFB, OH 45433-6543			
8a NAME OF FUNDING/SPONSORING ORGANIZATION AIR FORCE SYSTEMS COMMAND	8b. OFFICE SYMBOL (if applicable) PMREB	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-87-C-1491			
8c. ADDRESS (City, State, and ZIP Code) AERONAUTICAL SYSTEMS DIVISION / PMREB WRIGHT PATTERSON AFB, OH 45433-6503		10 SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO. 65502F	PROJECT NO. 3004	TASK NO. 10	WORK UNIT ACCESSION NO 84

11 TITLE (Include Security Classification)

Biomasscomp

12 PERSONAL AUTHOR(S) Robert L. Dawes

13a. TYPE OF REPORT FINAL	13b TIME COVERED FROM 87Aug18 TO 88Feb18	14 DATE OF REPORT (Year, Month, Day) 1988 September	15. PAGE COUNT 190

16 SUPPLEMENTARY NOTATION

This is a Small Business Innovation Research Program Report, Phase I

(over)

17	COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

BIOMASSCOMP is a project whose objective is to define and develop methods for automating the process of "reverse engineering" the brain for application to the development of intelligent sensors and controllers for avionic and other systems. What we have done in this project is to quantify and apply concepts that many neural network and cognitive science researchers have tacitly and qualitatively assumed to be at work in self-organizing systems. Our experiments have shown that these assumptions need to be much more carefully thought out.

During this Phase I SBIR project, we have defined, developed, and implemented an entropy-based scalar measure, DMORPH, of the common structure between two systems, as evidenced by (over)

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ( )UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☒ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Maj. James R. Johnson	22b TELEPHONE (Include Area Code) (513) 255-6453	22c. OFFICE SYMBOL AFWAL/AAOR

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted SECURITY CLASSIFICATION OF THIS PAGE
All other editions are obsolete

Block 16 (Continued)

Cys of Statement of Terms and Conditions--Release of Air Force-Owned or Developed Computer Software Packages will be furnished upon request to AFWAL/AAOR, Wright-Patterson Air Force Base, Ohio 45433-6543.

Block 19 (Continued)

measurement of signals from the two systems. By design, DMORPH reflects only the crosscorrelations between systems and not the intracorrelations within the separate systems.

DMORPH was applied to the input and output signals from various artificial neural network architectures to attempt to determine which networks, and which parameter settings within each, induced the greatest structural similarity between input and output signals after learning had taken place. Networks tested included a "drive reinforcement" network of Klopf, a "back propagation" network, and a network which learns by a method of Bienenstock, et.al. The surprising results provided new insights into the relationships between cognitive systems and their environments and into the essential distinction between neural networks as cognitive systems and neural networks as mere associative memories. For example, the initial tests of DMORPH have explained the interesting psychological tendency of an observer to always perceive the greatest degree of order in his observed environment when his knowledge is at a certain intermediate stage between total ignorance and complete understanding. Yet, the application of DMORPH to network signals has shown that simple correlation between input and output signals is misleading and inappropriate as a measure of quality in a cognitive system.

This research applies to the development and testing of real time autonomous learning systems suitable for application to problems of avionics sensor fusion, adaptive sensor processing, and intelligent resource management.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

BIOMASSCOMP PHASE I FINAL REPORT

1.   INTRODUCTION AND OBJECTIVES

This is a technical report of a six month Phase I research project supported by the U.S. Air Force Wright Aeronautical Laboratories (WPAFB, OH) under the Small Business Innovation Research Program. This report presents a complete account of our investigations of the subject pursuant to the objectives of the original SBIR proposal, and is organized according to the listing of those objectives in the proposal. In the pursuit of those objectives, we have not only achieved the implementation of a successful entropic index of the structural similarity of two systems, but we have also identified errors in our planned approach through the conduct of experiments that failed to con-clusively demonstrate the expected results. This report details both the successes and the failures, and the valuable information that we have learned from them.

As stated in our Phase I proposal, our objective was to demonstrate the feasibility of developing and applying an entropic measure of structural similarity of systems so as to obtain (in the follow-on project) an automated procedure for mapping the architecture of a living neural network into a machine. In order to accomplish this objective, our tasks were:

1.   Identify and develop a mathematical technique for the measurement and analysis of relative information content in the signals of a network,

2.   Identify and develop a mathematical technique for the parametric optimization of artificial neural network

models as measured by the combined relative information
content of a functioning hybrid network,

3. Identify the functional design of a multichannel bidir-
    ectional signal translator suitable for the realtime
    interface of a natural network on the multimicroelec-
    trode plate (MMEP) apparatus of G. Gross at North Texas
    State University (NTSU) to an artificial network,

4. Closely monitor and assist the ongoing work at NTSU to
    demonstrate the capability of extracellular electrodes
    in the MMEP apparatus to be used for the injection of
    localized potentials capable of stimulating activity in
    specific subnets of the cultured neural network,

5. Closely monitor and assist the ongoing work at NTSU to
    demonstrate the ability to "condition" the behavior of a
    natural neural network in culture through controlled
    stimulation.

Our principal achievement has been the definition and
algorithmic implementation of a scalar measure of structural sim-
ilarity of two systems, based on extensive time-series of state
measurements (signal vectors) from the two systems. This report
details that definition, and the FORTRAN source code of the
algorithm is included in an appendix. A series of experiments
with random data vectors containing varying degrees of
correlations demonstrates the behavior of the algorithm.
Moreover, these experiments predict an interesting psychological
tendency of an observer to always perceive the greatest degree of
order in his observed environment when his knowledge is at a
certain intermediate stage between total ignorance and complete
understanding.

The application of DMORPH to neural network architectures
has shown that our approach to using the structure measure to
improve the architectural design of neural networks by comparing

them to natural networks appears to have been flawed. Yet, the flaw is not one that might have been easily detected without a study of the results of the experiments (although the a posteriori explanation in the form of a suitable "gedankenexperiment" is simple enough), and its exposure has resulted in new and useful insights into the structural and functional principles of neural networks and other self-organizing systems. Those insights are discussed in the "Analysis" section of this report, and they will constitute the direction for our planned Phase II research.

1.1   Background

The outline of our argument is this: The objective of an intelligent system is to minimize surprise, or novelty, in its interaction with its environment in a manner that is consistent with its "mission". To this end, it builds predictive models of the world and stores these models in any convenient recording medium including, but not limited to, its own memory. A predictive model will be more carefully defined below in Section 7, but heuristically it is a transition operator which associates each sensory measurement with an empirically-based probability density for the perceptive effects of future observations. (Ho & Lee [4]) This description is further illustrated by Watanabe, who says ([10], p.142) "The existence of structure means that the knowledge of a part allows us to guess easily the rest of the whole."

The brains of humans and animals are not apart from the universe, but are parts of the whole. Their function, which we summarize with the verb, "to learn", is to adopt a form which, when explored by the animal through associative recall, allows it to guess what is going on in the rest of the universe and to adjust its behavior to minimize surprize subject to its mission.

The design of artificial neural networks and neurocomputers normally proceeds by using the best available data from the neuroscience community to build and test computational models of the components and structures of the brain. Models that work well are improved upon. Models that flop are filed under "experience". The BIOMASSCOMP project (originally described in Dawes [2]) was designed to speed up this developmental process by defining a computable, numerical measure of the quality of a neural network model. This measure estimates the degree to which two neural networks are producing signals that have the same structure. A low value of the measure means there is little similarity in structure between the two systems. A high value means the structures are strongly correlated.

The initial application of the structure function was visualized to work as follows: An artificial and a natural neural network would be connected as in Figure 1 by a bidirectional communication link called the "synthetic axon bundle". The signals emanating from the natural network would be demodulated with a pulse-rate demodulator and would constitute one of two signal vectors. The signals emanating from the artificial network would constitute the other signal vector. The structure function would be applied to these two signal vectors and a numerical "structural similarity" would be obtained. As the two networks adapted through their learning laws, we would expect to see changes in the similarity value due to the intrinsic self-organizational behavior of both networks, and we would expect to see this value stabilize asymptotically in the absence of external stimuli to either system. Then, if we made any adjustment to the architectural parameters of either system, we would expect to see the structure value change again and stabilize on a different value. We would infer that the higher value of the structure function was obtained with the better set of architectural parameters and we could therefore obtain further improvements in the architecture by adjusting the parameters in the direction of the highest structural similarity. Since the

design parameters of the simulated network are under the dynamic
control of the experimenter's computer program, it can automatic-
ally increment or decrement these parameters so as to drive the
value of the structure function to its highest value.
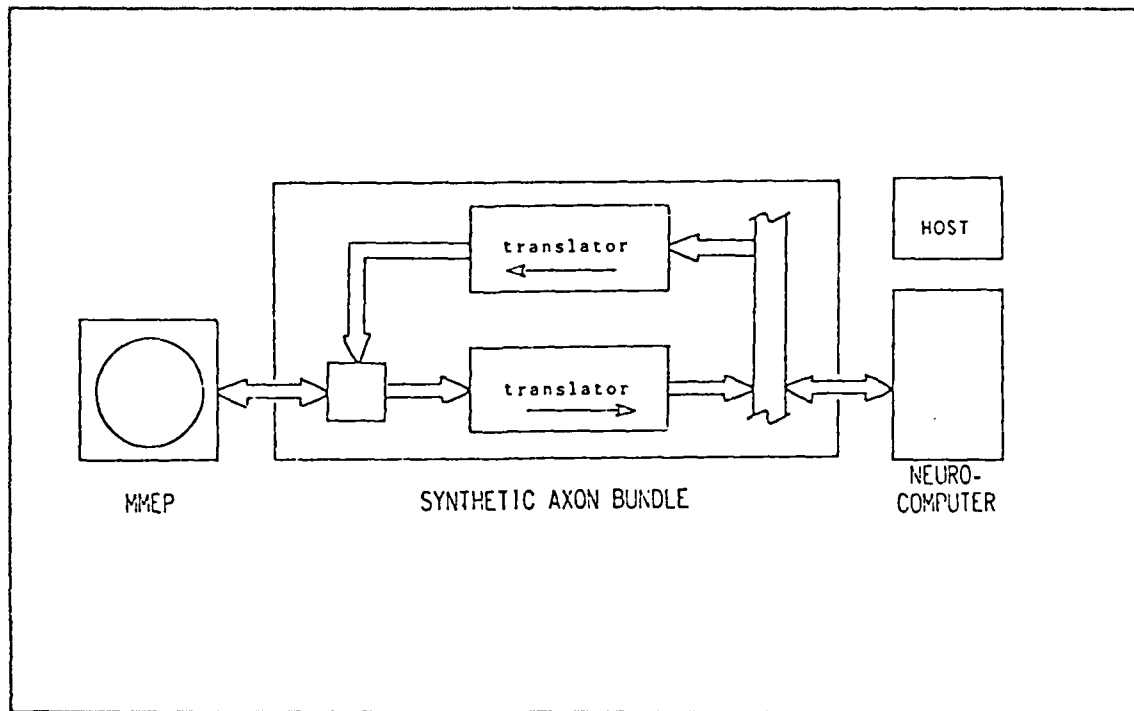
Figure 1.   The Hybrid Artificial/Natural Neural Network

Aside from the problems illuminated later by our experi-
ments, there are a number of difficulties that we could and did
anticipate. One of these is that the structure function is
difficult to compute, but we have made considerable progress in
that respect, as we shall demonstrate. More serious is the fact
that the performance of complex, nonlinear systems does not
always improve or degrade continuously as a function of their
design parameters. In particular, the performance may be subject
to bifurcations, catastrophes, and chaotic behavior as certain
parameters approach critical values. This means that the search
for improvement may have to be undertaken through stochastic

methods, such as simulated annealing, rather than by the more standard "hill-climbing" methods. Not the least of the problems is the realization of the bidirectional communication link between an artificial and a natural neural network, which relies on the successful development of a method for multiple-site stimulation of the natural network in the MMEP culture.

2.   DEFINITION OF THE STRUCTURE MEASURE

Our initial task has been to develop the measure of struc-
tural similarity of the two networks, based on the concept of the
Gibbs relative entropy function as described by Watanabe [10].
We now have a structure estimator (called DMORPH) running and
have tested it on actual pre-processed data from Prof. Guenter
Gross's laboratory at North Texas State University (cf., Appendix
A).

Any neural network which is worth its salt does at least
this one job well: It builds internal representations of
external events in such a way that an appropriate stimulus at a
later time will recover "a substantial portion" of the entire
representation.   Some have referred to this behavior as "self-
organization", but that is a seriously misleading phrase.
Consider the following homely analogy: Two politicians, Mr. A
and Mr. B, have widely differing world-views.   Mr. A sees
everything as either black or white, with nothing in between.
Mr. B maintains a complex set of concepts and classifications for
analyzing events.   A third politician, Mr. Z, has just died.

In terms of organization, as expressed by the entropy level
of his neural activation states, Mr. Z is extremely well-
organized, since the activation state of his neurons is a delta
function in time and space.   (This is to be distinguished from
his thermodynamic entropy which, while lower than that of a
warmer living brain, is still rather high.)   Mr. A is almost as
well organized, his activation state falling always into one of
two event categories, regardless of the facts of the external
world.   But Mr. B has the entropy of a very disorganized person,
since his mental state can be found in any of a large number of
configurations over time.   Unfortunately, if he is a member of
the "wrong" political party, Mr. B's ideas may reflect no corre-
lation whatsoever with reality in spite of their complexity.

Thus we see that organization, as measured by entropy, is not to be mistaken as an objective function for intelligence. There is no immediate monotonic relationship between the entropy (of the probability distribution) of the state of one's thoughts and the elusive quality we call intelligence. Yet the concept of entropy, properly applied, can help us to make this quality much less elusive.

The following sections detail the background and development of DMORPH. Additional technical details can be found in Jaynes [5] and in Watanabe [10].

2.1   Introduction to Entropy.

Entropy is a real-valued function whose domain is the set of probability measures on some given probability space. When a probability measure has a Radon-Nikodym derivative, $p(x)$, this is called the probability density function (pdf) of the probability measure, and in this case, the entropy of $p(x)$ is defined by

$$E(p) = -\int_X p(x) \log[p(x)] dx \ .$$

Whenever the probability measure is finite, i.e., there are only a finite number of events covering the sample space of x, then the integral above can be replaced by the sum

$$E(p) = - SUM\{ \ p_i \ \log(p_i) \ \} \qquad (1)$$

over the event sets with nonzero probabilities, p_i (which we shall refer to as the "nontrivial events of p"). It is not difficult to see that this quantity can range between a minimum value of zero, and a maximum value of $\log(N)$, where N is the number of distinct nontrivial events of p. The minimum value is taken when there is one certain event (in which case $N=1$). The

maximum value is taken when $p_i = 1/N$ for each nontrivial event of p.

In the following, we shall often speak of "the entropy of a system". This will always mean the entropy of the probability density function for the states that the system can take. Of course, the set of states for a given system is itself an abstraction and one may use different state spaces for different purposes. For our purposes, we are interested in the activation state of an ensemble of neurons, and not in their molecular kinetics.

The entropy of a neural system is an extensive quantity. That is, if the system is partitioned into two subsystems, there will be three possibly distinct entropies to deal with: Those of the two subsystems, and that of the whole system. Watanabe shows us how to relate these three quantities.

2.2   Entropic Structure

Suppose that we are given a system whose state space $\Omega = A \times B$, is represented as the Cartesian product of two subsystem state spaces, A and B. Suppose further that $x \in \Omega$ is distributed according to the pdf $P(x)$. If we write $x = (u,v)$, where $u \in A$ and $v \in B$, then we can obtain in the usual fashion the marginal probabilities of u and v as

$$P_A(u) = \int_B P(x)\, dv$$

and   (2)

$$P_B(v) = \int_A P(x)\, du$$

We can now obtain a **second** pdf on Ω as follows:

$$Q(x) = P_A(u)\, P_B(v) .$$

The random vectors u and v are _independent_ if and only if $P(x) = Q(x)$, by definition.

Now, given any two pdf's P_1 and P_2 on a single state space, Ω, J.W. Gibbs has defined the function,

$$G(P_1, P_2) = SUM_i [P_{1i} \log(P_{1i}/P_{2i})] , \qquad (3)$$

and has proved that it is always nonnegative, and that it vanishes if and only if $P_{1i} = P_{2i}$ for all i . It fails to be a metric on the space of pdf's on Ω, in part because it is not symmetric (although that is easily remedied), but we need not go into much more detail than this.

In the special case in which P_2 is derived from P_1 as Q was derived from P above, we can now define the _structure function_ $J_P(A,B)$:

$$J_P(A,B) = G(P,Q) .$$

The notation on the left stresses the fact that the state of the original system Ω is distributed by the pdf P, which is the only pdf in sight, and that each partitioning of the system into factor spaces A and B produces the nonnegative number $J_P(A,B)$ which depends on P and on the two marginal probabilities that fall out of the state space factorization. These two marginal probabilities have their own entropies, $E(P_A)$ and $E(P_B)$, and it can further be shown that

$$J_P(A,B) = E(P_A) + E(P_B) - E(P) \geq 0 . \qquad (4)$$

The proof is in Watanabe [10]. Because of this equation, the structure function is also referred to as the "excess entropy" generated by the assembly of two systems into one.

Note that the structure function vanishes if and only if the subsystems are statistically independent, i.e., if and only if their joint pdf (P) is the product of their separate pdf's. This in turn implies that there is no (pairwise) correlation between any component of $u \in A$ and any component of $v \in B$, although there may well be internal correlations within u and within v. The converse is false (lack of pairwise correlation does not imply independence), but the contrapositive is, of course, true: If crosscorrelations are nonzero, then the structure function will be strictly positive.

We have defined a normalized version of the structure function, called DMORPH, which is constrained to take values between 0 and 1, regardless of the dimensions of the state spaces and regardless of the (finite) number of primitive events which partition the state spaces. Thus, we have

$$\text{DMORPH} = J_P(A,B)/M_P(A,B), \tag{5}$$

where

$$M_P(A,B) = E(P) - \text{Max}[E(P_A), E(P_B)] .$$

The normalization divisor, $M_P(A,B)$, is only our best current estimate of an upper bound for the structure function. We have not proven that is a supremum. It is obtained by supposing that the smaller of the two systems (say, B) is completely correlated with a subsystem of the larger, in which case the "excess entropy" is the difference between the entropy of the whole and the entropy of the larger subsystem. This un-rigorous argument is supported by numerical experimentation and might be made into a proof with a little more effort.

The normalized structure function, DMORPH, is the tool which we have used in our experiments to measure the relative similarity of structure between two systems based on vector samples of

signals from the two systems. Computation of DMORPH relies on
the ability to estimate the three constituent entropies, and this
is not an easy task. Perhaps the most significant contribution
of this work is the development and implementation of this
algorithm.

2.3   Computation of the Structural Similarity (DMORPH)

In order to estimate the entropy (relative or otherwise) of
a system, it is necessary to obtain an estimate of the probabil-
ity density function for the state-vector of the system. In
order to account for both spatial and temporal correlations in
the joint PDF, the state-vector must be sampled and held over a
time interval which is long enough to span the coherence of the
system. Since the long-term memory of a neural network is
supposed to maintain temporal coherence over the lifespan of the
network, it will clearly not be possible to sample, hold, and
process the full quantity of data needed to characterize the
system!

Instead, it will have to suffice to use a time window which
is long enough to cover the short-term dynamics of the network,
i.e., its "impulse response". With such a window, the resulting
estimated PDF will reflect the short-term memory (STM) of the
network, including both the neuronal transitions and the network
effects, but can only hope to reflect as much of the long-term
memory dynamics as are evident within the sample window. These
may not be insignificant. According to Klopf [6] the coherence
needed to obtain storage of long-term memories in animals is on
the order of about three seconds.

Consider now the problem of estimating the PDF for the
signal vector (including a number of time samples) of the system.
Traditionally, this would be done by partitioning the ranges of

the samples into "event bins", and accumulating a histogram.
The entropy would then be estimated by computing

$$E = SUM_i[(N_i/N)*log(N_i/N)], \qquad (6)$$

where N_i is the number of occurrences of the i-th event, and N is
the total number of trials in the experiment. For a vector with,
say, 8 components and a time-window of, say, 32 samples per trial
(10 samples/sec for 3.2 sec) and a partition of the range of each
of those 256 variables into, say, 8 levels -- that comes to n =
8**256 possible events! This is clearly beyond the capacity of
available computational methods.

The estimation technique known as the Maximum Entropy Method
(MAXENT, cf., Jaynes [5]) overcomes these problems by constrain-
ing the pdf's of interest to lie within certain limited classes
of functions. For example, they may seek the pdf of maximum
entropy among all those pdf's whose mean and covariance are equal
to the sample mean and the sample covariance. Under these and
similar constraints, the solution may be found by the method of
Lagrange multipliers. Although these methods have been applied
to the study of living neural networks [9] we assert that the
effort is futile. The pdf of the signal vector of a neural
network is typically extremely complex, as befits a system which
by design extracts and stores millions of similarity clusters of
data which it finds in its sensory inputs. Thus the signals from
such systems will of the essence be "mega-modal". But the MAXENT
distribution determined by the first two statistical moments (on
an infinite domain) will be unimodal, namely the multivariate
Gaussian. Since it is precisely the fine structure of the signal
density that we are interested in, and not the textbook
statistical parameters, we need a nonparametric method which
reflects only those constraints imposed by our measuring
instruments.

In the end, our methods for understanding and reverse engineering the brain will look very much like the method that the brain uses for understanding and reverse engineering its sensory environment. Such an "eternal golden braid" will make the recursive tangles of Godel, Escher, or Bach look like mere children's toys. In the following description of our method for estimating structural similarity, the reader is invited to observe that <u>the design of the computational methods may hold more clues to the design of intelligent systems than will result from their application</u>.

We are approaching the problem in the following way. For computation of the DMORPH function THREE entropies must be computed. We can reduce that to just ONE entropy via the following argument.

We argue that <u>any</u> partitioning of the sample space into events prior to the collection of data imposes an unwarranted bias on the resulting entropy measurement. For example, the use of 7 threshold levels (defining 8 events) on a real-valued measurement must of necessity define two regions which are infinite in extent, and the placement of these thresholds presumes some knowledge of where the bulk of the measurements will lie. Therefore, for our computations, we use only the a-priori knowledge of the computational resources at our command to select the NUMBER of event-bins for each component of the sampled data[1] ; and we then adjust the BOUNDARIES of these bins (i.e.,

----

1: In truth, even this strategy imposes hidden a-priori constraints. It assigns an arguably unwarranted priority to numerical contiguity within events. Why, for example, should numerically contiguous events be preferred by nature over, say, a partitioning in which events are defined by the value of the third significant digit of an octal representation of the measurements? The answer is reasonable and straight-forward: Our measuring instruments have inertia and this results in an unavoidable time-averaging of results. Thus, the definition of events by contiguous ranges of measurements automatically incorporates this smoothing constraint.

the thresholds that separate them) during data collection so that
the number of observed events per bin is the same (+/- 1) for
each bin.   This results in a "tiling" of the n-dimensional event
space (n = product of sample-vector dimension times the number of
time-samples per trial) into equiprobable events.
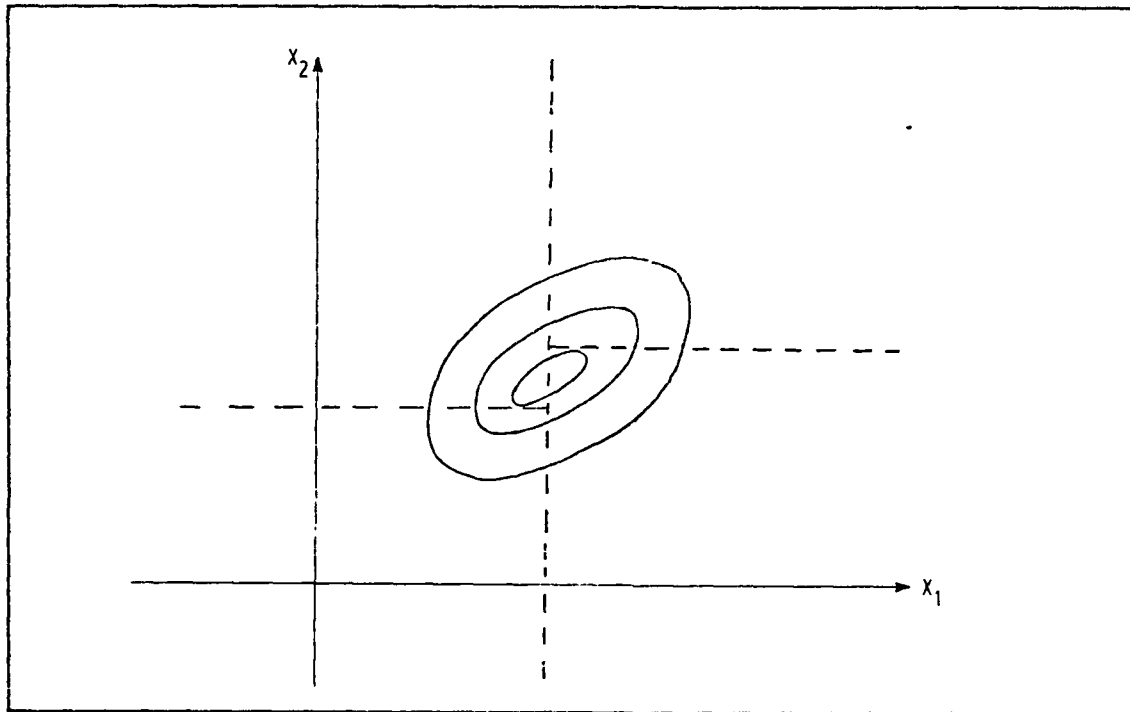
Figure 2.   Binary Tiling of 2-D Sample Space

The actual procedure is illustrated for a two-dimensional
random vector as follows.   (See Figure 2.)   Tiling of the subsys-
tem sample spaces is easily accomplished using a sort routine on
the components of the sample vector.   We are currently partition-
ing each component into only two equiprobable events because of
the computational limitations.   (There is a big difference
between (2)**256 events and (3)**256 events.)   Thus we look
first at component i=1 of the sampled data and use a sort
routine to find the median, where we locate the single threshold
for the first component.   Then, for all sample vectors whose

first component lies below the threshold, we find the median value of their second components. This becomes the first of TWO thresholds for the second component. The second threshold is obtained by finding the median of the values of the second components whose first components lay above the threshold. This results in four "tiles" which partition the two-dimensional samples into equiprobable events. If the sample vectors were three-dimensional, there would be four thresholds on the third axis, and the seven total thresholds would partition the three-dimensional space into 8 equiprobable events.
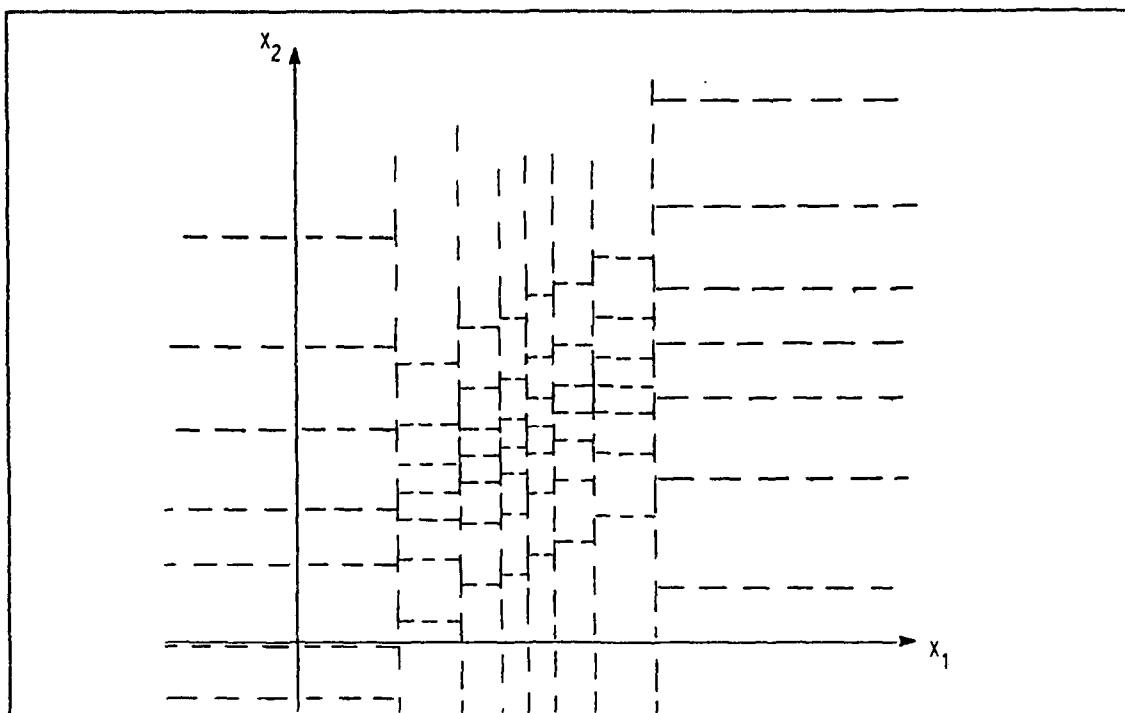
Figure 3. Tiling of 2-D Sample Space with 8 Events per Segment

This tiling of the space is (in the limit of large numbers of thresholds on each axis) equivalent to obtaining a PDF for the data, in that the reciprocal of the volume of each nonvacuous tile is proportional to the probability of finding the state of the system in each unit volume within that tile. Although the tiling is obviously dependent on the order in which the axes are

selected, we conjecture that the differences become insignificant
as the number of thresholds on each axis increases. It is
instructive to observe that the tiling is a constructive repre-
sentation of the probability measure which has the histogram PDF
for its Radon-Nikodym derivative. The measure (more precisely,
the inner Jordan content) of any set of states is obtained by
counting the number of tiles within the set, the same as would be
done if the tiles were defined more customarily as unit hyper-
cubes. This is more easily seen when the number of events per
segment is larger than 2, as in Figure 3. That is, whereas the
unit tiling which is used for building a normal histogram repre-
sents the (translation invariant Lebesgue) measure associated
with the uniform PDF, the tiling derived from the data represents
the (in general, non-translation invariant) measure associated
with the actual PDF of the data.

Now, if we had only ONE entropy to compute, there would be
nothing to it, because our tiling guarantees that the sample
frequencies are uniform, and the entropy of a uniform distribu-
tion over n events is the maximum possible: $\log(n)$. But we are
measuring the entropies of two presumably coupled systems, having
n and m events, respectively, and there are THREE entropies in
question, namely the two entropies of the separate systems, and
the entropy of the composite system. We are at liberty to tile
the sample spaces of the two subsystems any way we like. But
once the events in the subsystems are defined, then the events in
the composite system are determined as the product space of the
two subspaces. The resulting entropy estimate $E(P)$ for the
composite, therefore, must be computed by the usual formula (6).
It will be somewhat less than its maximum value $(\log(m*n))$,
according to the degree of crosscorrelation, or structure,
between the subsystems, and the relative (excess) entropy will be

$$J_p = \log(m) + \log(n) - E(P) > 0. \qquad (7)$$

Page 17

From this, it is then easy to compute DMORPH using equation (5).
One observes that because the tiling maximizes the entropy
estimate for each of the subsystems, DMORPH measures only the
crosscorrelations which exist between them and is unaffected by
any changes in the internal organization of one which are not
reflected by corresponding changes in the other. This is also
confirmed by our experiments.

Listings of the program (DTEST) which computes DMORPH and
its associated tilings and entropies are included in Appendix B.
Experiments showing the performance of DMORPH are described in
Section 2.4, and the experimental configurations and their
resulting graphs are shown in Appendix C.

We conjecture that this method of event-boundary adjustment
can form the basis for a learning law for neural networks which
maximizes the information content of internal representations of
external events. This will be investigated further in Phase II.

2.4   DMORPH Characterization Experiments
2.4.1   Description

The DMORPH experiments were performed to test and verify the
performance of the algorithms which construct equiprobable event
tilings of the two subsystem sample spaces, which compute the
entropy of the composite system, and which compute the normalized
structure function, DMORPH.

In the first set of experiments (1 through 4), random
vectors X and Y of dimensions 2, 4, 6, and 8 were generated.
Their components were uniform in the interval $[0,1)$.[2]   These

---

2:   The tiling algorithm worked so well that it immediately
     detected a serious fault in our random number generator,
     which we subsequently replaced with an algorithm from
     Abramowitz & Stegun.

experiments determined the running time for the algorithm and demonstrated the relationship between the dimensions of the subsystems and the amount of data which was needed to stabilize the entropies and the structure value, DMORPH. A sample plot showing the entropies (upper three curves) and DMORPH (lower curve) as they evolve with additional trials of the experiment is shown in Figure 4. Similar plots, together with the experimental configurations are found in Appendix C to document the characterization experiments.
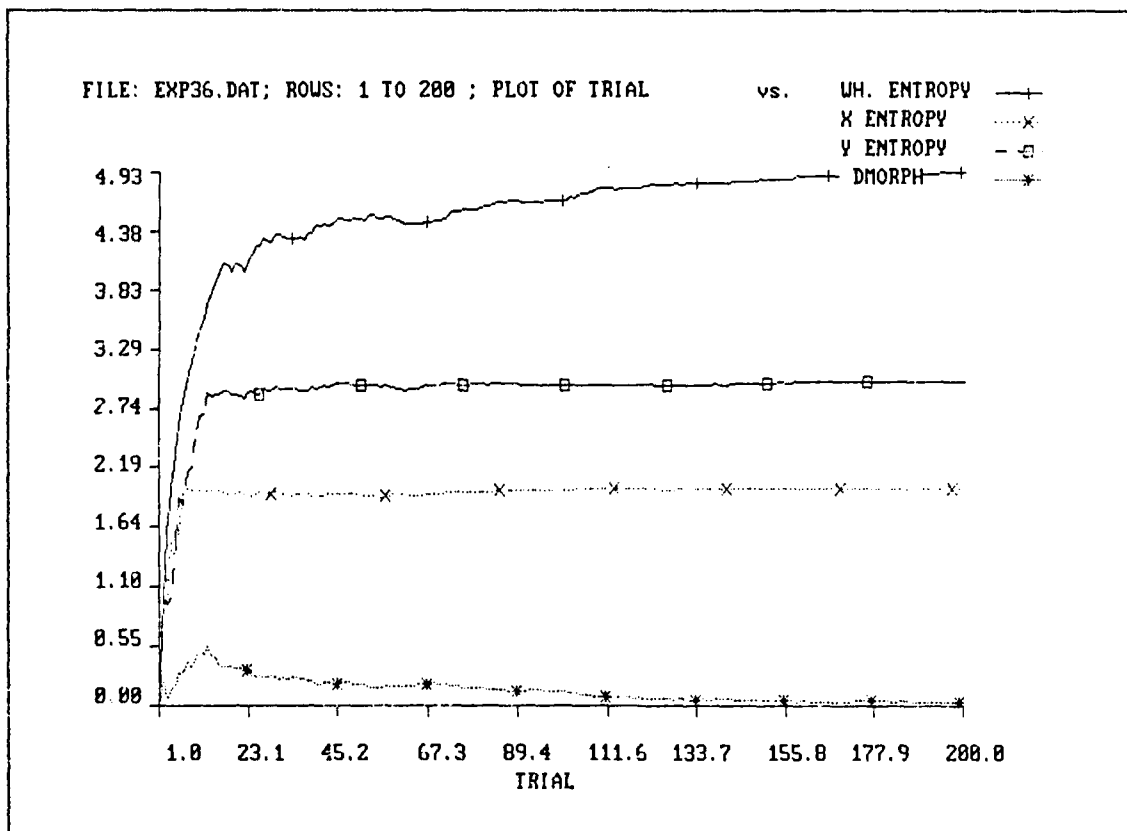

FILE: EXP36.DAT; ROWS: 1 TO 200 ; PLOT OF TRIAL vs. WH. ENTROPY X ENTROPY Y ENTROPY DMORPH

Figure 4. DMORPH Experiment for Dim(X)=2, Dim(Y)=3 Pseudorandom Data without gross correlations.

In the second set of experiments (5 and 6), intracorrelations were introduced into one or both of the random vectors X and Y to determine whether the measured structural similarity between X and Y were indeed independent of these intracorrelations.

In the third set of experiments (7 through 35), crosscorrelations were introduced between X and Y to determine their effect upon the value of DMORPH and to evaluate the normalization divisor to see whether it is close to the theoretical maximum of the relative entropy.

2.4.2   Results

The graphs in Appendix C illustrate the results of the DMORPH characterization experiments.

Graphs of experiments 1 to 4 show that as the dimensionality of the systems increases, the quantity of data needed to obtain stable estimates of the entropies and, hence, of DMORPH also increases.   This is because the total number of distinct events defined by the tiling algorithm on a subsystem of dimension N is 2^N, and before the subsystems can possibly exhibit their maximum entropy, the number of samples per bin must be somewhat larger than unity.   The tiling algorithm insures that within each subsystem, the events will be defined so that no matter how much data is taken, the greatest difference between the number of samples assigned to any two event-bins will be plus or minus one. (Exceptions occur whenever certain sample values occur multiple times, which prevents insertion of a threshold to separate them.) But whenever that difference is a sizeable fraction of the total number of data samples per bin, the entropy will be substantially below its theoretical maximum.   In particular, when the first sample is taken, the distribution of samples in the event bins looks like a delta function, so the entropy always starts at

zero.   After that, it climbs toward its theoretical maximum
(since the event-bin boundaries are being adjusted during data
collection), which is $\log(2^N)$. We convert the logarithms to the
base 2, so that the theoretical maximum entropy for any of these
systems is just the dimension of the system.

Note that the value of DMORPH in these graphs begins at zero
and then rises to a maximum value before falling back toward
zero.   The reason that it falls back toward zero, of course, is
because there is very little crosscorrelation between the
components of X and of Y.   (The residual makes a nice measure
of the quality of the pseudorandom number generator.) But the
fact that it rises to a maximum showing some "false" structure
before adequate data is collected (see Figure 4 again) leads to
some interesting comparisons with the way people learn about
their environment.   It seems to say that when we are confronted
with a totally structureless system to observe, and we begin to
collect data on it, we will first be convinced that the system is
structureless; then we will begin to see patterns; but as the
data becomes statistically complete all the patterns disappear.
It also confirms the wisdom of carving out low-dimensional
analytical tasks, because with the really big problems (e.g.,
Neural Network theory, or Artificial Intelligence) the amount of
data which one is likely to obtain during the attention-span of
the average funding agency will most likely lead one to make
grandiose claims of great discoveries which are doomed to
evaporate when the data are more complete.

Subsequent experiments, described below, will show the same
qualitative behavior, except that when there really is some
crosscorrelation between the subsystems, the "false" structure
then gives way to a nonzero asymptotic value.

In the next set of experiments, numbered 5 and 6 in Appendix
C, intracorrelations were introduced (e.g., $X_1 := X_2$) and the
results show that DMORPH is impervious to such deception.

Finally, crosscorrelations were introduced (e.g., $X_2 := Y_2$, and $X_2 := X_2 + Y_2$), and these show that the structure function is properly sensitive to them. Many experiments were performed, showing, e.g., the structure between the input random vector and the output random vector for various simple matrix transforms. Finally, when Y is made equal to a subvector of X, DMORPH rises almost to unity, showing that the normalization divisor is, if not precisely correct, quite adequate to measure the relative structure between two systems without being biased by the dimensionality of the problem.

3.   ADAPTIVE STRUCTURE OPTIMIZATION

In order to test the applicability of the structure measure
to the control of neural network designs, it is necessary to
implement one or more typical neural network designs and simulate
the interaction of that network with the biological network.
With such a simulated interaction, it is possible to treat the
randomly generated (and naturally generated) input signals as
being representative of the natural network structure, while the
output signals represent the artificial network structure.

Originally, it was our intent to implement these trial
networks on the MassComp MC5700 at the Biosciences Laboratory at
North Texas State University, which has direct access to the
signals emanating from a living culture of several hundred
mammalian neurons (see Appendix A). However, it would not be
possible to perform the pulse-rate demodulation and the DMORPH
tiling in real time, and there is as yet no capability for the
artificial network model to talk back to the natural network with
any form of stimulation. Therefore, we opted to take simulta-
neously recorded data from multiple channels in digital form and
to perform the experiments off line at our own facilities.

In the following sub-sections, we describe the models which
we have implemented for these experiments. The details of the
experiments and their results are described subsequently in
Section 6 of the report. All of our network model software was
implemented under our proprietary dynamical system simulator
package, SYSPRO^{TM}. This has allowed us to program the published
versions of these models into a flexible simulation module with
minimal duplication of effort. In all cases, it is only
necessary to produce a SYSPRO primitive system which computes the
transfer function and the learning algorithm of the subject model
and to link it into a possibly minor modification of our SYSPRO
composite network model as the replicated node. The specific

interconnect graph is specified at run time through the system
initialization instructions.

3.1   The Back-Propagation Model

The back-propagation model which we used is the one which is
described in Rumelhart and McClelland [7].   This required almost
no programming effort, since it is a model with which we have
extensive experience and which we include as a sample network in
our commercial neural network simulator package.   The network was
configured as a 4-3-4 feedforward network (including direct links
from the input layer to the output layer).

The transfer equations for the processing elements are given
by

$$y_j = \sigma (\text{SUM}_i\ z_{ji}\ x_i\ ;\ B,\ C),$$

where $\sigma(A;B,C)$ is a sigmoidal function of the first argument,
with values ranging between 0 and 1, whose maximum slope, C,   is
attained at A = B.   Except for the ability to control the value
of the slope (C) at the desired threshold (B), this function is
the commonly used "logistic" function:

$$\sigma(x;b,c) = 1/[\ 1 + \exp(-4c(x-b)\ ].$$

The learning law is modified only so that those factors of the
update equations which can be computed "locally" are computed
within the neuron model, and those which require information at
the network level are computed by the network model.   (Our
simulator protocol facilitates the assembly of system models
written in heirarchical fashion, but it imposes the discipline of
using only data which is available to subsystems through the
input terminals and the local state vector.)

3.2   The Bear-Cooper-Ebner Model

The BCE model is based on the description by Bear, Cooper, and Ebner [1] of a learning algorithm attributed to Bienenstock. This learning algorithm is partly Hebbian and partly anti-Hebbian in that each synaptic weight learns in proportion to the presynaptic activation, but the proportionality constant may be positive or negative according to the relationship of the current post-synaptic activity to the recent **average** of the post-synaptic activity. That is, if the recent activity has been high, but the current activity is lower than the average activity, the synaptic weight will be reduced. If the recent activity has been low, then almost any post-synaptic activity will be greater than the average activity and will cause an increase in the synaptic weight. The Bienenstock law is,

$$dm_j/dt \;=\; \varnothing(c,\bar{c}) \; d_j \;,$$

where m_j is the j-th synaptic weight, d_j is its presynaptic signal, c is the neuronal output signal (in the linear region), and $\bar{c}$ is the average of c over a recent time interval.

We implemented the $\varnothing$ function (see Figure 5) as a spline of a parabola on the left and an exponential learning curve on the right of the crossover point, $\varnothing_M$. We implemented the neuronal transfer function more generally than is described in BCE, so as to include a sigmoid nonlinearity at the output (the same logistic sigmoid used above in the back-propagation model), rather than to assume operation in the linear region. This necessitated a decision on the interpretation of c in the learning law, and we chose to interpret c as the neuronal output, rather than as the postsynaptic activation.
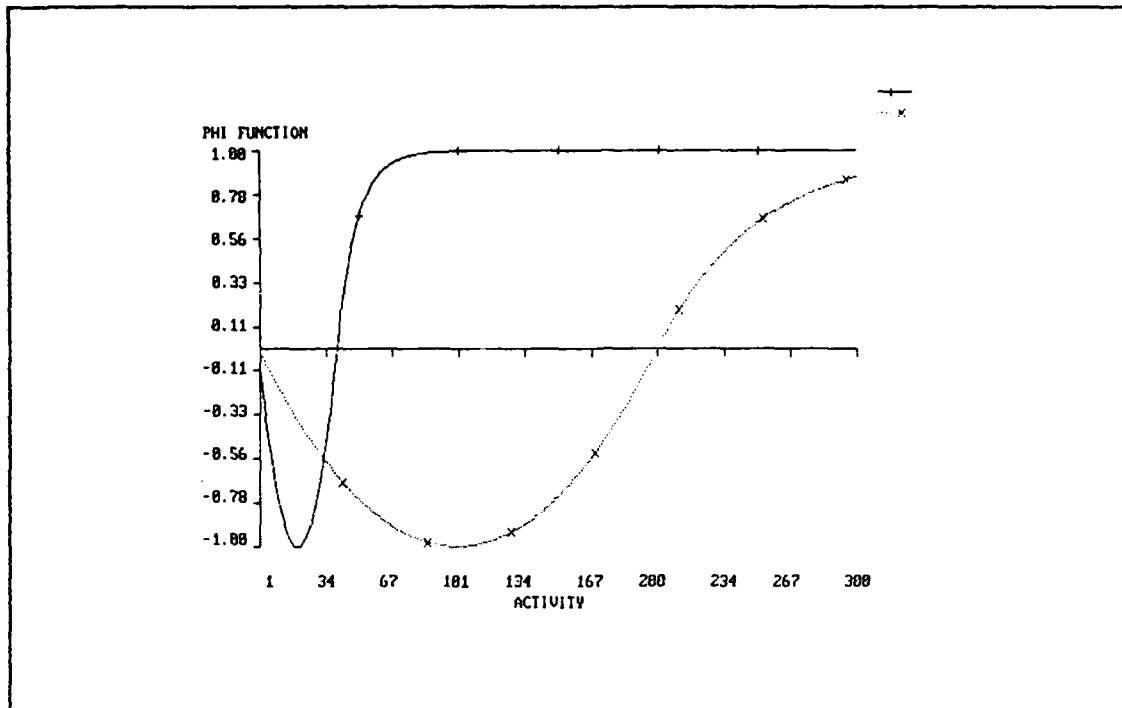
Figure 5.   The Bienenstock $\emptyset$ Function (two versions).

3.3   The Klopf "Drive-Reinforcement" Model

We implemented the drive-reinforcement model in accordance with the description given in Klopf [6].   In order to control the learning rate, we augmented the learning algorithm with a GAIN factor, which in effect scales the area under the learning rate constant curve (the curve determined by the constants, c_j, in Klopf's report).   By setting GAIN = 0, we could turn learning off at any time so that the tiling operation of DMORPH would have a time-invariant segment of the network's signals to work with, i.e., one in which the structure was not changing during the tiling operation.   Also, since the GAIN factor was included, we chose to implement the learning rate constants, c_j, so that their sum is unity.   This then treats the c_j vector as a weight vector for a weighted average of the prior history of the

synaptic efficacies. The relative magnitudes of our values for c_j were (almost) the same as used by Klopf.

When the Klopf neurons were connected into a network, we decided not to make any of the synapses non-plastic, since inside the network it is unlikely that the distinction between conditioned stimuli (CS) and unconditioned stimuli (US) could be made a-priori, and in any case our time constraints did not allow such fine-tuning.

--- This page is mostly blank ---

4.   DESIGN OF THE BIDIRECTIONAL AXON BUNDLE

In this section we describe the proposed technique for
constructing an interface between a living tissue culture of
active mammalian neurons and an artificial neural network which
is hosted in a general purpose computer.   The design is based on
the laboratory setup in the neurophysiology laboratory of Dr.
Guenter W. Gross at North Texas State University and on his
proposed apparatus for localized stimulation of that network.

First we present a brief description of the NTSU laboratory
apparatus, which is more thoroughly described in the Appendix.
After that, we describe the status of the work being conducted at
NTSU and at Southern Methodist University to achieve the local-
ized stimulation of the culture network.   Finally, in the third
subsection following, we describe the functional design of a
bidirectional interface, called the Synthetic Axon Bundle, which
will enable the culture network to influence and be influenced by
the signals in an artificial neural network.

4.1   Description of the NTSU-Gross Apparatus

Professor Gross's laboratory apparatus is described briefly
here and illustrated thoroughly in Appendix A.   The multimicro-
electrode plate (MMEP) on which the culture is maintained is
described first, followed by a description of the recording
chamber design.   The digital processing system is illustrated on
page A-18.

Signals from the neural culture are amplified and patched
into the data acquisition and control processor of the Masscomp
MC5700 computer, where they are converted to digital form,
typically at a 30 kHz per channel sample rate.   The digital
signal is filtered for A/C hum and is then processed for burst
detection (pages A-24 to A-29).   Signals at various stages of

processing can be selected for display on the color monitor (page A-25).

The signal monitoring apparatus is supplemented by an auditory monitor, which codes each electrode's activity into tone bursts at a frequency that is unique to the source electrode, and by an LED display which provides visual cues to the activity on each electrode. These were originally designed as "PR enhancement instruments" (where PR stands for Public Relations), but they have proven to be valuable intuitive aids. The human ear can detect patterns and correlations in the data that would go completely unnoticed on a strip-chart recording.

Not shown in the hardware description of Appendix A is a limited capability for stimulation of the cultured network. This is described in greater detail below. First, we describe the design of a Synthetic axon bundle which presumes the availability of a suitable stimulation apparatus.

4.2   Functional Design of the Synthetic Axon Bundle

The purpose of the Synthetic Axon Bundle is to provide the communication link between the natural mouse neural network (MNN) in culture and the Artificial Neural System (ANS) being simulated on the MASSCOMP. That is, it's function is to

(1) Modulate the signals that are output from the ANS so that they can be used to stimulate the MNN and

(2) Demodulate the signals that are recorded from the MNN so that they can be used as input to the ANS.

Figure 6 illustrates the basic functional design, and the following description provides some of the details.
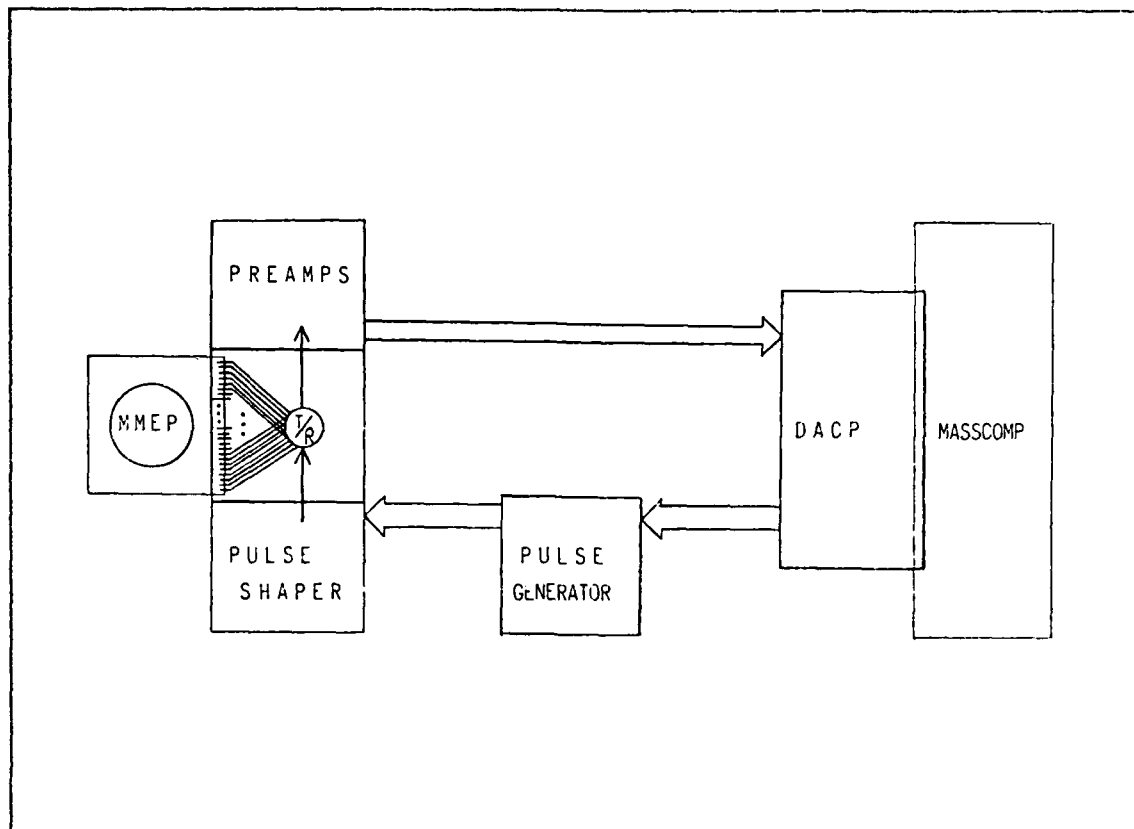
Figure 6.   Design of the Synthetic Axon Bundle

ANS -----> MNN

It is believed that the output of an ANS should represent
some sort of information transfer, either by its effect on the
modification of synaptic weights or the interpretation of what
this neuron firing means (the recognition of some pattern, say).
Therefore it is necessary to modulate each ANS output into a
spike train to be used to stimulate the neurons of the MNN which
are in close proximity to a particular electrode.   There is no
requirement that the processing clocks of the two neural systems
be on the same time scale.   The only requirement that exists is
that the data from the ANS be modulated in such a manner that the
MNN finds it to be "stimulating".   Some experimentation will be

needed early in Phase II to verify that the following proposed
design will result in a real time stimulus of the MNN.

The simulation time increment, Δt, of the ANS model will be
set so that the data produced by it each Δt can be modulated and
used as a stimulus over the next Δt seconds for the MNN. Of
necessity, there will be a Δt second time lag between the data
output from the ANS and the stimulus being applied to the elec-
trodes of the MNN. Since we expect to perform temporal sampling
as well as sampling across the electrodes this should not be a
problem in our search for cross-system structure.

The amplitude of the voltage spikes generated to drive the
MNN will be consistent with the amplitudes observed in the MNN.
The frequency used will be proportional to the signal amplitude
out of the ANS.

MNN -----> ANS

The signals recorded from the MNN were collected on one of
multiple micro electrodes in Dr. Gross's laboratory. The data
used in many of the experiments performed during the Phase I
effort was compressed using the following processing algorithm.

Eight simultaneous channels of data were collected at a
data rate of 30,000 Hz.

This 30,000 Hz. data rate is then reduced to 500 Hz. by
saving only the maximum absolute value of each disjoint
and contiguous set of 60 data values on each channel.

The dynamic range of these data values is further reduced
by comparing the data value with a threshold and replacing
it by a 1 if the data value is greater than or equal to
the threshold or replacing the data value by zero if it is
less than the threshold.

Finally, a 16 point rectangular filter (with unit weights)
is applied to a sliding window of the data so that the
data which is input to the ANS is an integer between 0 and

16, inclusive. This data is to be interpreted as Pulse Repetition Frequencies (PRF) for the neurons which are being recorded on each electrode. The following table gives the range of PRFs for each of the possible 17 data values.

$M(t) = 0$			PRF	< 31.25	Hz.
$M(t) = 1$	31.25	$\leq$ PRF	< 62.50	Hz.
$M(t) = 2$	62.5	$\leq$ PRF	< 93.75	Hz.
$M(t) = 3$	93.75	$\leq$ PRF	< 125.0	Hz.
$M(t) = 4$	125.50	$\leq$ PRF	< 156.25	Hz.
$M(t) = 5$	156.25	$\leq$ PRF	< 187.50	Hz.
$M(t) = 6$	187.50	$\leq$ PRF	< 218.75	Hz.
$M(t) = 7$	218.75	$\leq$ PRF	< 250.00	Hz.
$M(t) = 8$	250.0	$\leq$ PRF	< 281.25	Hz.
$M(t) = 9$	281.25	$\leq$ PRF	< 312.50	Hz.
$M(t) = 10$	312.50	$\leq$ PRF	< 343.75	Hz.
$M(t) = 11$	343.75	$\leq$ PRF	< 375.00	Hz.
$M(t) = 12$	375.00	$\leq$ PRF	< 406.25	Hz.
$M(t) = 13$	406.25	$\leq$ PRF	< 437.50	Hz.
$M(t) = 14$	437.50	$\leq$ PRF	< 468.75	Hz.
$M(t) = 15$	468.75	$\leq$ PRF	< 500.00	Hz.
$M(t) = 16$			PRF	≥ 500.00	Hz.

This algorithm has several defects, but it also has the important advantage of its mere existence. Thus, we were at least able to run experiments on genuine digitized data from the MMEP, but the interpretation of results must be qualified by the effect of the following problems.

First, the data collected on any single micro electrode is known to be the result of firings of multiple neurons. These signals should really be separated rather than lumped together.

Second, the threshhold used to declare a signal present on the channel (electrode) is not changing over time and therefore the false alarm rate is not constant on the channel.

Third, the data is collected at an extremely high data rate but the high data rate features of the data are not exploited in the signal processing algorithm at all. Why waste all the magnetic storage?

Fourth, the definition of bursting for the channel is
limited to 17 discrete values rather than taking on any value on
the positive interval from zero to the sampling rate.

A set of signal processing algorithms which addresses the
majority of these problems have been formulated by Martingale
Research Corporation and supplied to Dr. Gross, but they have not
yet been implemented due to lack of funds at NTSU for support of
student programmers. The design specification for these algo-
rithms were presented in (Dawes and Collard [3]) and are repro-
duced in Appendix D.

5. PROGRESS WITH STIMULATION AND CONDITIONING

In order to utilize our structure function for improvement of neural network designs in a real time interactive experiment, it is required that the natural and the artificial neural networks be connected for bidirectional communications. Part of the technology to accomplish that is described in the preceding section and is called the "Synthetic Axon Bundle". It is basically little more than a modulator/demodulator (MODEM) which translates the signals from a form suitable to their source to a form suitable to their destination. But the specific component which injects the signal into the MMEP has not been specified or tested yet.

What is needed are the following capabilities:

1. A multichannel pulse generator whose signal output characteristics are subject to computerized control individually by channel according to pulse amplitude and pulse rate.

2. A localization of the voltage gradients within the MMEP so that gradients capable of inducing depolarization into neurites are limited to the vicinity of the active electrode.

The first requirement is not a big problem, but the second is more difficult to satisfy. At present, the input signal is applied between the selected electrode and the metallic bezel which surrounds the recording area (see page A-9). This results in depolarization of an estimated 10% to 40% of the neurons in the culture. Anything which is more selective will have to be based on the bipolar excitement of adjacent pairs of MMEP electrodes, and this will require some redesign of the preamplifier boards.

A team of electrical engineers under the direction of Prof. Lorn Howard at the Electrical Engineering Department of Southern Methodist University is working on the stimulation problem. Dr. Gross at NTSU presently has a spike-signal generator connected to the MMEP which is capable of injecting a signal into a single electrode at a selectable pulse width and rate. But without sufficient control to be able to simulate bursting, he is unable to demonstrate any form of conditioning of the network.

6. EXPERIMENTS PERFORMED AND THEIR RESULTS

6.1 Back-Propagation Experiments

We ran some simple experiments using an eleven neuron feed-
forward network which learns by the "back-propagation" algorithm
to simulate the communication of a noncontrollable neural network
with a controllable one. The feedforward network is the 4-3-4
network shown in Figure 7. Its four input signals consisted of
raised sinusoids of various amplitudes and phases and its output
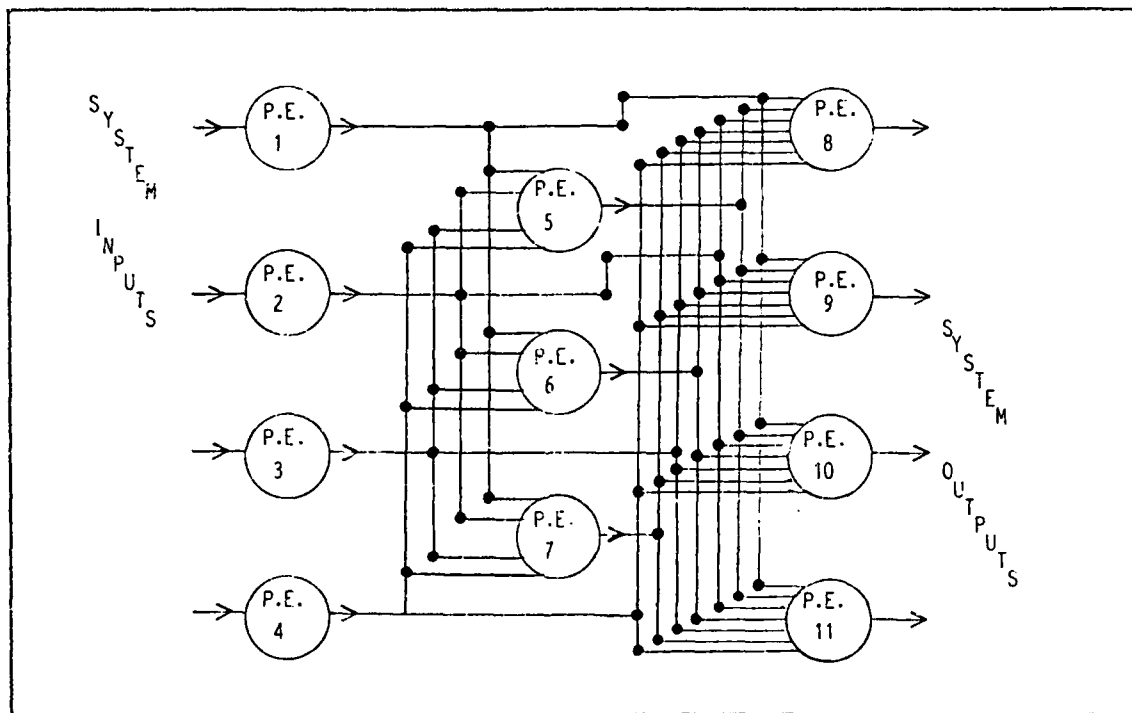is determined by the weights and biases of the processing
elements.

Figure 7.   The 4-3-4 Backpropagation Network

To test the structure function on this network, we ran two
experiments. Both experiments used the same input vector
consisting of the four raised sinusoids, and both began with a

random initialization of the weights of the network. With
learning turned off, the network was exercised for 1500 seconds
of simulation time (at 10 state updates per second) to obtain
baseline structure with the randomly initialized weights. Then
the learning was turned on and the "desired" output was set equal
to the input vector. Convergence was fairly rapid, and the
output took the qualitative appearance of the input, with
distortions appropriate to the nonlinearities of the sigmoid
function. Then learning was turned off and the network was again
exercised to obtain the post-learning structure.

In the first experiment (BP4 #1,2, p. C.34-35), the baseline
value of DMORPH was 0.356 prior to learning and 0.336 afterward!
The structure actually declined after learning. Obviously there
was an error in our procedure.

Ordinarily, we would not report on the many experiments in
which we have identified procedural errors, but in this case, the
error was especially instructive and it supplied us with an
important clue to the role of causality in neural transductions.
That clue is taken up again and discussed at greater length in
Section 7. For now, we merely point out that in the first
experiment, we sampled both the input vector and the output
vector immediately upon presentation of each new input to the
network. Consequently, the signal that was present on the output
terminals was the one that was left over from the previous input.

The second experiment was the same as the first, except that
the inputs and outputs were sampled on the half-second instead of
at each whole second of the simulation clock. This allowed five
simulation cycles for the input, which only changes at the begin-
ning of each whole second, to propagate through to the output.
In this experiment the pre-learning structure was 0.446 (BP4 #3,
Page C.36), and the post-learning structure was 0.515 (BP4 #4,
Page C.37).

Not only did the post-learning structure show an increase over the pre-learning structure, but the pre-learning structure as determined by the lagged sampling (0.446) was higher than the pre-learning structure of the first experiment (0.356). This is because in the first experiment, the structure reflected the correlation that exists between one sample of a sinusoid and the network-transduced image of another sample approximately 1/20 cycle into the past. In the second experiment, the structure reflected the correlation that exists between each input sample and its own transduced image.

6.2 BCE Experiments

The BCE experiments were originally planned to parallel the foregoing BPE experiments for a small network whose learning law is due to Bienenstock (described above). After implementing and testing a network of neurons using that learning law, however, we observed a phenomenon that led us to discard our planned experiments, at least for now.

What we observed was that if a constant nonzero input was supplied to one synapse of a BCE neuron, the synaptic weight, and consequently the neuronal output, quickly reached a periodic state. The period of the oscillation was directly proportional to the length of the window used for the sliding-window average of the postsynaptic activation. That is, a long window produced a low frequency oscillation, while a short window produced a high frequency oscillation. The length of the period is of the same order of magnitude as the length of the sliding window. The graph in Figure 8 shows this behavior.

In retrospect, it might have been worthwhile to go ahead and perform the DMORPH experiments on this kind of network, but there was not sufficient time. We report these results as a matter of interest regarding possible future use of the model.
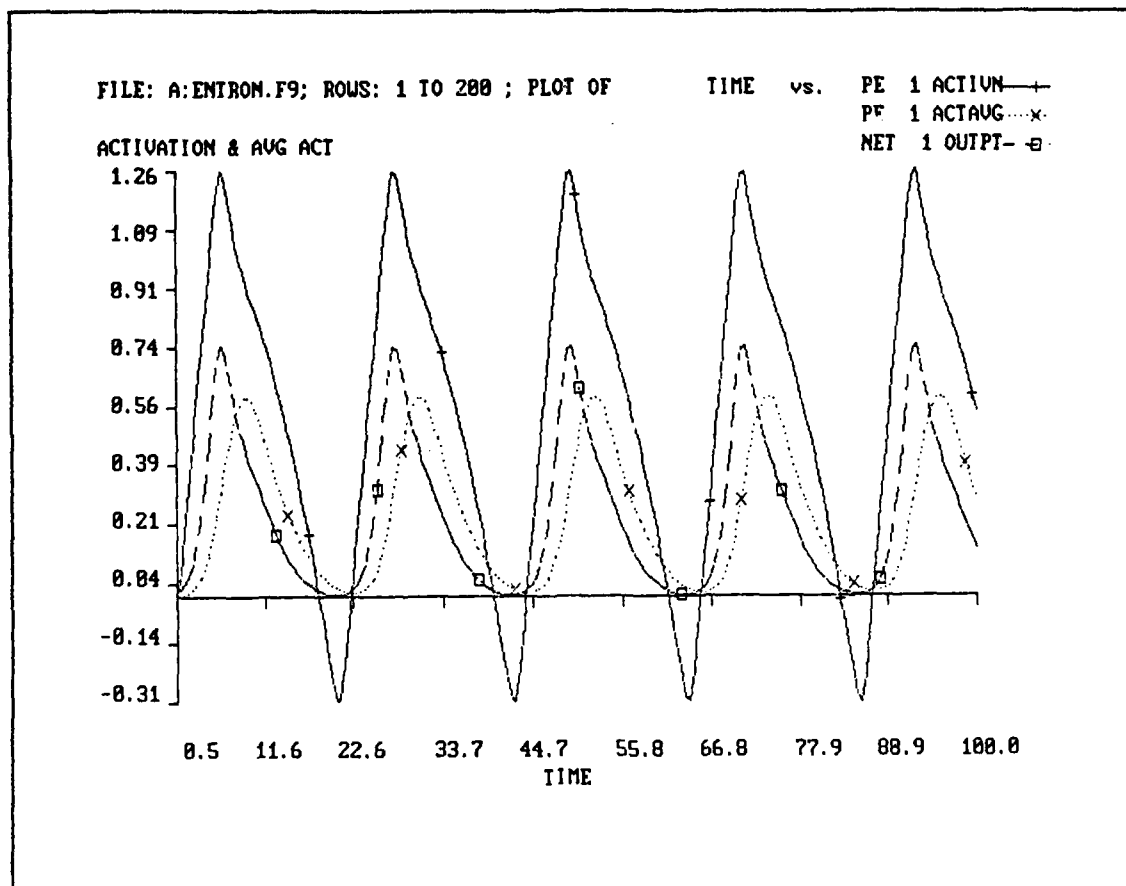
Figure 8. Oscillatory BCE Response to Unit Step Input

6.3 Drive-Reinforcement Experiments

In late December, we received Klopf's report [6] on his work with the modified differential Hebbian learning law of Sutton-Barto, which he calls the Drive-Reinforcement model. We were able to implement a small network of neurons with the D-R learning law in a couple of days, and we ran a set of experiments similar to the BPE experiments, but on a four-neuron network. In this series of experiments, each neuron received a separate input, and each neuron's output was sampled. Thus our structure function was evaluated on a four-by-four system.
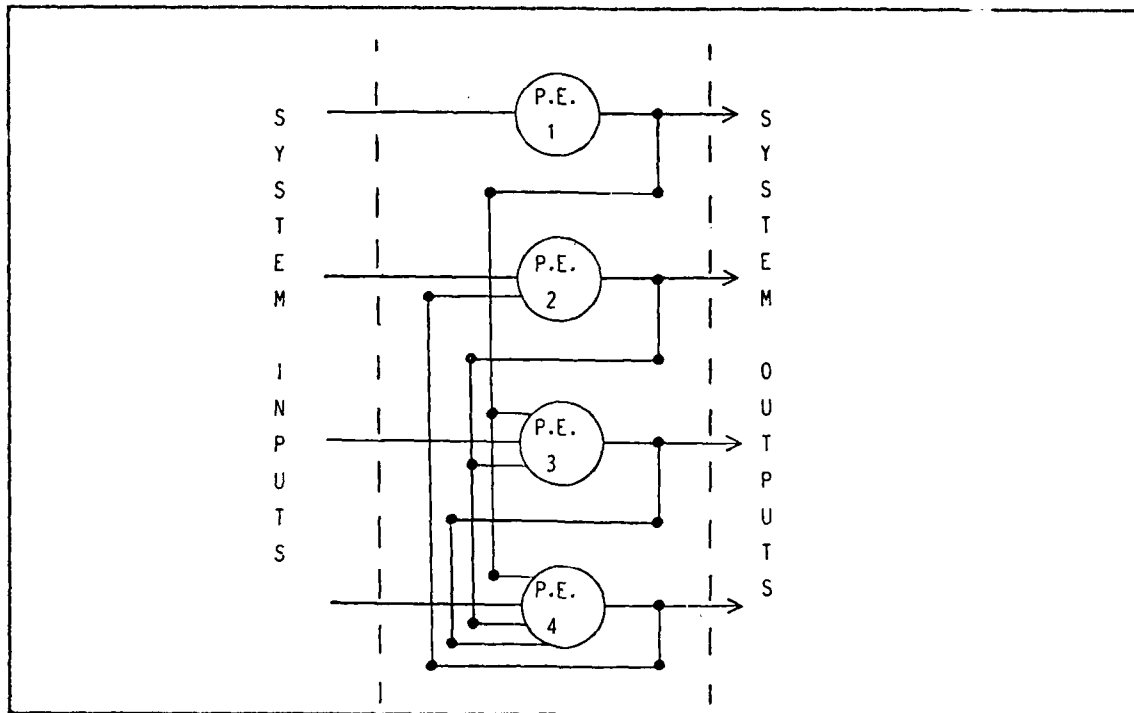
Figure 9.   The Drive-Reinforcement Network

The experiments were designed to measure the effects of both the learning algorithm and the architectural parameters on the structural similarity between the input vector to the network and its output vector.   To determine the effect of the learning algorithm, we began with a random initialization of the synaptic weights on a four-neuron network connected as in Figure 9.   This interconnection incorporates two feedback loops, which helps to randomize the latency between onset of signals at the various inputs to each neuron and thus guarantees that the drive-reinforcement learning algorithm sees the necessary delays.   We did not hard-wire any synapses, since in the network setting it is not clear that any synapse may know a-priori that it will be receiving the unconditioned stimulus (US).   The components of the C vector (learning rate at delays of 0.5, 1.0, 1.5, 2.0, 2.5 sec) were established at approximately 1/10 of the values suggested by Klopf so that the area under the sliding "C" window would be

unity.   In particular, we chose   C = (.4, .25, .2, .1, .05).   By
setting the GAIN parameter at 10, we could scale these back up to
approximate Klopf's settings, or we could alter their values
directly.

The four input signals were the same raised sinusoids as
were used for the Back Propagation experiments, except that their
values only changed every three seconds in this experiment.
Since the Drive-Reinforcement neuron cycles only twice each
simulation second, this gives time for the signal to propagate
around the feedback loops before the input is changed.   Other-
wise, the procedure was the same as before.   Learning was turned
off to obtain a baseline input/output structure.   Then learning
was turned on until the synaptic weights had undergone signifi-
cant change (but not long enough for any weight saturations to
occur).   Then learning was turned off and the network was
exercised with the learned weights to obtain the post-learning
input/output structure.

The pre-learning structure with random weights was 0.480 and
after learning it was 0.412 (Page C.38,39).

A second experiment was run in which the values of C(1) and
C(2) were reversed.   This was an attempt to determine the effect
of architectural adjustments on the post-learning structure.   The
pre-learning structure, of course, did not change, because this
architectural change does not affect propagation when the learn-
ing is turned off.   After learning for the same length of time as
before, the post-learning structure was 0.391, slightly worse
than before (Page C.40).

6.4   Other Experiments.

One of our experiments was designed to look into the impli-
cations of time and causality.   That experiment was performed by

partitioning five of the 8 channels of actual action potentials
from the MMEP into an X signal of dimension 4 and a Y signal
of dimension 1. We then extended the Y signal along the time
axis so that its first component was sampled at the same time as
all four of X's components were sampled, and its next three
components were taken at three subsequent time values, so that Y
is also a four-vector. Our reasoning is that if one or more of
the X signals were to correlate with anything in Y, it would
only be seen at a later time. The situation is shown in Figure
10.
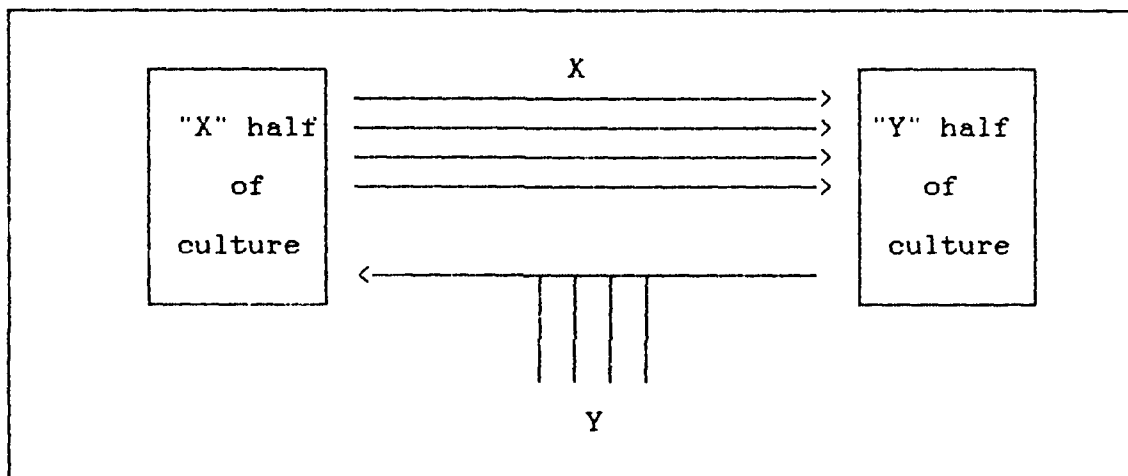
Figure 10.   Sampling Diversity in Space and in Time

The result was a DMORPH value of 0.1, which is far less than was
achieved with the simultaneous sampling in which both X and Y
were 4-vectors (DMORPH = 0.3). Similar experiments on different
segments of the MMEP data show qualitatively similar results.

We also programmed a network of Grossberg Avalanche neurons,
which learn by a form of the basic Hebb rule. We used them to
verify the basic performance of one of the avalanche architec-
tures, but due to time constraints, we did not employ these
models in any of the structure function experiments. This kind

of experiment on these models will have to await Phase II research.

The results of the initial tests have been to illustrate that the measurement of the relative entropy between two systems is not a simple matter, as one might already guess from reading the MAXENT literature (cf. [5]). Nevertheless, we have obtained an algorithm which operates well within the time and memory constraints imposed by a PC computing environment when the vector dimension of the combined random signals from two systems is 8 or less. This is adequate for determining the viability of the proposed method.

7.   FINDINGS AND ANALYSIS

In the Back Propagation experiments, the structure increased after learning, but in the Drive-Reinforcement experiments, it actually went down afterward. Moreover, in the MMEP signal studies we saw more structure in the simultaneous samples than in the samples showing both spatial and temporal diversity.

These results seem ambiguous at best. In the case of the Back Propagation experiments, the structure could hardly fail to increase after learning, since the "desired" output was known to be strongly correlated with the input and the BPE network learns to produce the desired output rather well. In the Drive-Reinforcement experiments, the failure of the structure to rise after learning is puzzling. It is clearly an indicator that the D-R learning law does not necessarily enhance correlations between input and output signals, but then it was not designed to do that, at least not directly.

We devoted considerable effort to our attempts to understand these results and to appreciate their implications for both the design of learning laws and the optimization of architectural parameters. Eventually, thanks to insistent questioning by our student assistant, Mr. David Boney, we happened upon the following "gedankenexperiment", which shows very clearly that one cannot naively infer that the best neural network is the one which generates the greatest value of DMORPH between its input vector and its output vector.

Suppose that the artificial network were constructed with an array of "bypass valves" corresponding to each of its inputs, and that these valves served to proportionately disconnect the network from its inputs and reroute those inputs directly to the output terminals. Then, as we turned the dials we would see the structural similarity between the input signal vector and the output signal vector improve dramatically toward its theoretical

maximum, and our inference would lead us to conclude that the best network architecture was the one that was not there at all!

The problem is not a fault with the structure function, but rather with its application and the inferences drawn from it. We hasten to point out that what we have done in this project is to quantify and apply concepts that many neural network and cognitive science researchers have tacitly and qualitatively assumed to be at work in self-organizing systems. Our experiments have shown that these assumptions need to be much more carefully thought out.

What, then, is the measure of a cognitive system? At the beginning of this work, we dismissed naive self-organization -- at least as it might be measured by information-theoretic entropy, since that clearly favors mental crystallization. We now seem driven to dismiss, or at least to severely qualify, the placement of any value on the network's introduction of cross correlations between its inputs and its outputs. Parrots are only amusing for a short while, and networks which merely associate "desired" outputs with selected clusters of inputs hardly know what constitutes a surprise, much less do they have any hope of developing an appropriate response to one.

The defect in these models, insofar as they attempt to represent basic elements of cognitive systems, is that they pay too little attention to the fundamental role of time and causality. Even the conditioning models discussed by Klopf [6], which explicitly account for the temporal ordering of events and the temporal gradients within excitations, and which do an admirable job of modeling drives, reflexes, and even the act of generalization, will probably not emerge into cognitive systems through the blessings of mere complexity.
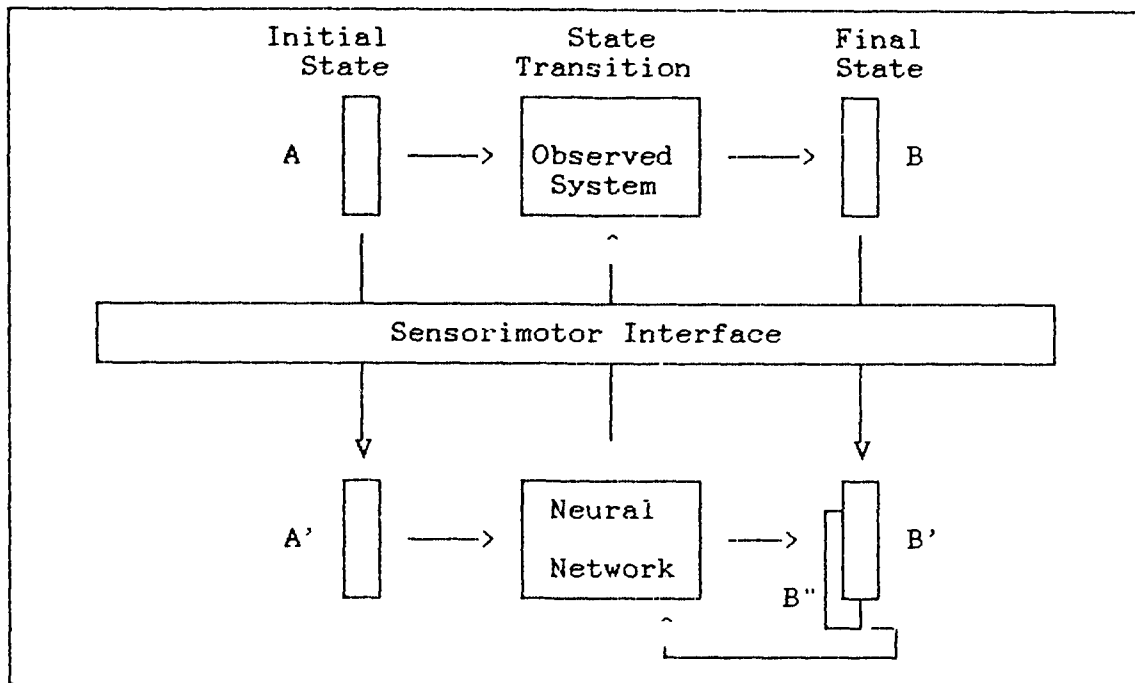
Figure 11.   The Neural System Design Problem

Our experiments with time and causality discussed above have
led us to the following description of a learning system, which
pays special attention to its relationship to its environment.
Figure 11 illustrates the situation in a manner which is intended
to be especially significant to mathematicians who may have
studied category theory.   Category Theorists are sometimes known
among mathematicians as "arrow chasers".   In this case, we are
interested in the fact that there are three paths leading from
the initial state, A, of the observed system to the final state,
B'/B", of the neural system.

One path, which we call the LL path (for Lower Left),
represents the sensors mapping the initial state A of the
observed system into an initial internal representation A'.
Then the neural network transforms that representation into a
final state B" (which may be only an infinitesimal time, dt, away
if we think of these as differential systems).

Another path, the UR path, represents the observed system evolving from its initial state A to its final state B, possibly under the influence of inputs from the outside world. Then the sensory network maps that final state into the internal representation, B'.

The third path, called SN (for SiNuous), represents observation of the initial state, followed by action of the neural system upon the observed system through motor controls, which produces a controlled final state B. This final state is then observed, producing the internal representation, B'.

For the moment, we shall ignore the SN path, and ask how the network can make TFDC (an acronym known by category theorists to mean "The Following/Foregoing Diagram Commutes"). That is, how can the two representations, B' and B", be made to coincide, so that both the LL and the UR paths produce the same result? The answer is that it must build within itself a state transition operator which, when composed with the effect of the sensors, produces the same result. This is the meaning of "learning" in a sense which makes essential use of the dynamics of the universe, including that small part of it called the neural system. It absolutely must employ a means of comparison between the two representations, B' and B". We have illustrated that comparison by the juxtaposition of two state boxes for B' and B", and an arrow which returns a control signal to the current transition operator of the network; but we do not mean to imply literally that there must be two such "slabs" within the network together with an explicit "metric" between them. That may be the case, but it may also happen that the sensory map and the cognitive prediction converge on the same layer to produce a disturbance away from homeostatic equilibrium at precisely those locations harboring the pieces of the distributed transition operator which need correcting.

It happens that we have, in the process of analyzing these experiments, constructed both a neural model (a totally novel one at that) AND an appropriate architecture to imbed it into, which may achieve these goals. The model is highly preliminary, and since it was conceived in the final days of analysis and report-writing, it must be reserved for further development in the Phase II research.

Now, let us return to the intriguing SN path. This path is the only thing which distinguishes the cognitive system from a mere cork on the currents of the universe. There is a technique in the theory of the Monte Carlo method which is called "importance sampling", in which the experimenter salts the random data with certain rare events which are known to have an important effect on the simulations, but which are too rare to just sit and wait for in truly random data. In a similar fashion, a cognitive system requires repetition in order to ferret out the associations and the invariants which it needs to build its internal models. But novel events, by definition, do not present themselves at frequent intervals. Therefore, the cognitive system must have a way to salt its observations, and it does this by manipulating its environment to repeat the novel event or to inspect it from a different angle so that the tentative learning (called a hypothesis) can be tested and adjusted before it evaporates. This is a necessary function of the SN path. Essentially, it exists as a means to "salt" the experience of the network and improve the efficacy of learning. But it can also serve to drive B' toward B" in the event that learning fails to drive B" toward B'.

It is tempting to suggest that the entropic structure analysis, via DMORPH, can once again be brought to bear by using it to compare the structural similarity between the two representations, B' and B", but this would not be appropriate. DMORPH only applies to random vectors and ergodic stochastic processes, whereas the comparison between B' and B" which is used for

learning must be almost instantaneously computed on a sample-by-sample basis. It now seems that the two structures between which DMORPH might be expected to find similarity are the state transition operator of the observed system and the state transition operator of the neural system. In the case of an artificial neural network, this operator is available in the form of the matrix of synaptic weights and the associated nonlinear transfer characteristics, but they must be treated as random operators (cf., Skorohod [8], for the linear case) or else the entropy will collapse to zero. If this seems difficult to carry out, it is no doubt far easier than the other half of the problem, which is to estimate the transition operator of the observed system (except in the most simple and controlled experimental configurations). It may well be, as we have alluded to above, that in order to measure the structural similarities in such complex systems, our measurement instrument will have to contain the cognitive equivalent of the system it is trying to measure.

In closing this analysis, we would be remiss if we did not acknowledge the role of experiment in the development of our own cognitive models of cognitive systems. We have developed a tool, the DMORPH algorithm, which provided corrective inputs to our models both by its success in measuring structural similarity in random signals, and by its success in showing that such structure is not relevant in the ways that we had presupposed when we initiated this project. The paths to understanding are often more sinuous than direct. We had intended that DMORPH should be applied to the reverse-engineering of the brain in a direct, gradient ascent assault. Instead, it is in the design and verification of DMORPH that we found unexpected clues to the organization of cognitive systems. We expect that this sinuous path will now be more productive than the original plan.

8.   CONCLUSIONS AND APPLICATIONS

We are confident that cognition is simply not to be under-
stood in isolation from the essential interaction of the cogni-
tive system with the environment which it learns to comprehend.
No neural network, however complex, will exhibit cognition if it
is relegated to passive observation of its environment. The
conclusion here is perhaps the only solid confirmation of a pre-
conceived idea which we had prior to beginning this work: It is
that significant progress with neural networks cannot be expected
without the maintenance of close ties with biology.

This work shows the potential value of stochastic structure
analysis in the design and improvement of neural network models
and it is clear that in six months we have only begun the process
of testing and analysis of the various network designs. Now that
the software tools are available, the structural analysis
deserves to be carried out in a thorough and organized fashion on
many of the existing network architectures to determine whether
and to what degree their learning algorithms record and reproduce
the structure in the signals that they observe.

Although the DMORPH algorithm may not be applicable as
originally planned to an automatic architecture mapping scheme,
it clearly has an immediate utility for the evaluation and
improvement of neural network learning algorithms and transfer
equations. Our intention is to refine the algorithm (its sort
routine could be made faster and less sensitive to multiplicity
of data) and commercialize it as a utility to our commercial
neural network simulation package, SYSPRO^{TM}. Furthermore, it is
clear that we can employ both SYSPRO and DMORPH for neural
network design and evaluation for the benefit of the Air Force
and other government agencies who must compare designs and
determine their applicability to their needs.

--- This page is mostly blank ---

9. REFERENCES

1   Bear, M.F., Cooper, L.N., and Ebner, F.F., "A Physiological Basis for a Theory of Synapse Modification", Science, vol. 237, 3 July, 1987.

2   Dawes, R.L., "BIOMASSCOMP: A Procedure for Mapping the Architecture of a Living Neural Network into a Machine", Proc. IEEE First Int'l Conf. on Neural Networks, San Diego, June, 1987, p. III-215.

3   Dawes, R.L., and Collard, J.E., "Functional Design of a Realtime Multisensor Signal Processing Package for Action Potentials of Neurons in Culture", Technical Report No. MRC-NTSU-86-1, Martingale Research Corp., August, 1986.

4   Ho, Y.C., and Lee, R.C.K., "A Bayesian Approach to Problems in Stochastic Estimation and Control", IEEE Trans. Automat. Contr., vol. AC-9, pp. 333-339, Oct. 1964.

5   Jaynes, Edwin T., "On the Rationale of Maximum Entropy Methods", Proc. IEEE, vol 70, no. 9, Sept., 1982.

6   Klopf, A. Harry, "A Neuronal Model of Classical Conditioning", AFWAL-TR-87-1139, WPAFB, OH, October 1987.

7   Rumelhart, D., and McClelland, J., Parallel Distributed Processing, MIT Press, 1986.

8   Skorohod, A.V., Random Linear Operators, D. Reidel, Boston, 1984.

9   Victor, J.D., and Johannesma, P., "Maximum-Entropy Approximations of Stochastic Nonlinear Transductions: An Extension of the Wiener Theory", Biol. Cybern. 54, 289-300, 1986.

10 Watanabe, S., Pattern Recognition: Human and Mechanical, John Wiley & Sons, 1976.

APPENDIX   A

THE NORTH TEXAS STATE UNIVERSITY NEUROSCIENCE LABORATORY
OF
PROF. GUENTER W. GROSS

APPENDIX

i

Dr. Guenter W. Gross   CENTER for NETWORK NEUROSCIENCE   North Texas State University
                                                                                                      as of May 1988: University of North Texas

_____

1. MULTIMICROELECTRODE PLATES (MMEPs) IN USE

A 1
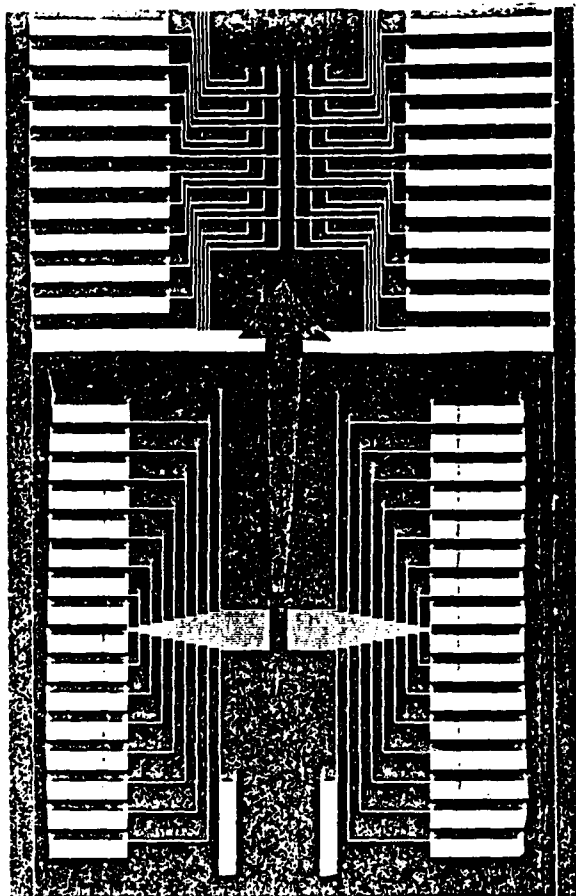
Fig. A 1. MMEP 1. The lower picture shows
the entire 5 x 5 cm plate with conductors and
amplifier contacts. The center area is enlarged and
shown on top. The 36 electrode are arranged in
6 rows and 6 columns with 200um and 100um spacing
respectively. Each conductor is 10 um wide and 1um
thick. These plates were fabricated in 1976 free of
charge by the Siemens Corporation in Munich, Germany.
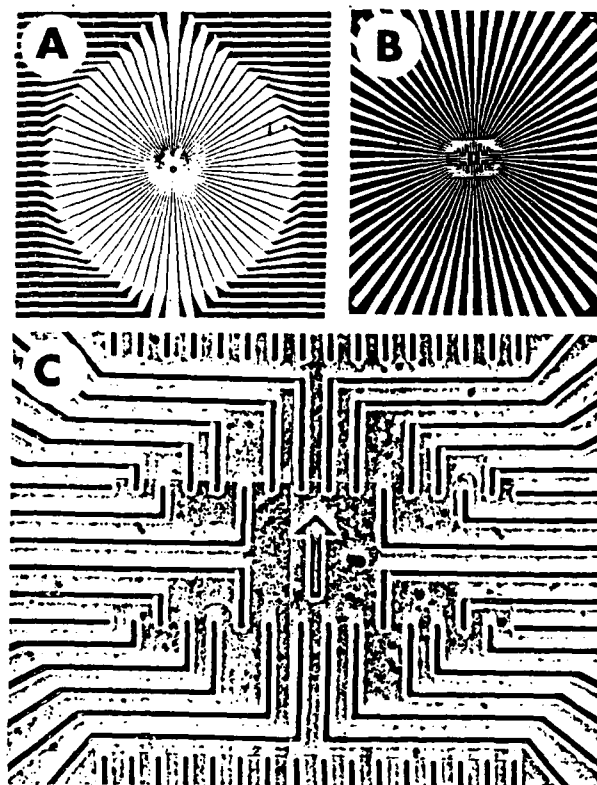
FIG.**A2.** Multimicroelectrode plate with 64 indium-
tin oxide electrodes. (A) 5cm x 5cm x 1.2 mm
plate showing overall arrangement of conductors.
(B) Central region of MMEP showing 0.5 mm x 0.5
mm recording matrix and radial conductors. (C)
Phase contrast micrograph of the recording area
(conductor width: 10 μm; column spacing: 40 μm;
row spacing: 200μm).

A4

A4 Electrode shunt capacitance (curves) in picofarads on left ordinate and shunt impedance at 1 kHz (straight lines) in megohms on the right ordinate as a function of insulation layer thickness. Calculations were carried out for maximum and minimum conductor areas resulting from different conductor lengths situated under the saline pool. Solid lines represent extreme values when a circular open culture chamber is utilized (27 mm diameter). Dotted lines result from the maximum conductor area under saline when a 25 mm x 42 mm closed culture chamber is used (Fig. 1). (Relative dielectric constant = 4.)
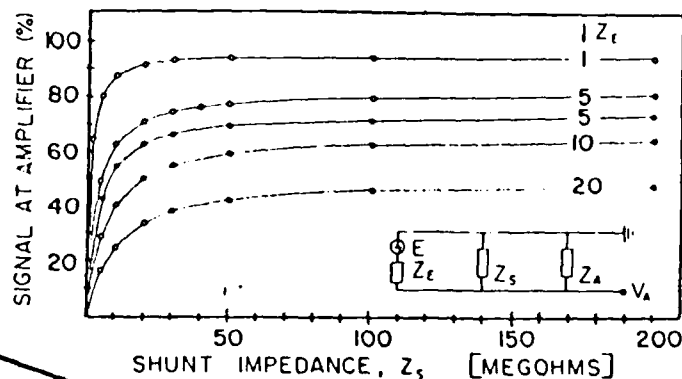
A5 Percent of signal (E) seen by electrode tip reaching amplifier as a function of shunt impedance Z_s and electrode impedance Z_e (1 kHz). Serious signal attenuation occurs at electrode impedance above 5 MΩ and shunt impedances below 30 MΩ. Only small improvements in electrode performance can be expected from increasing the shunt impedance above 50 MΩ. Curves were calculated with an amplifier input impedance (Z_a) of 20 MΩ (open circles) and 15 MΩ (solid circles, with Z_e equal to 5 MΩ).
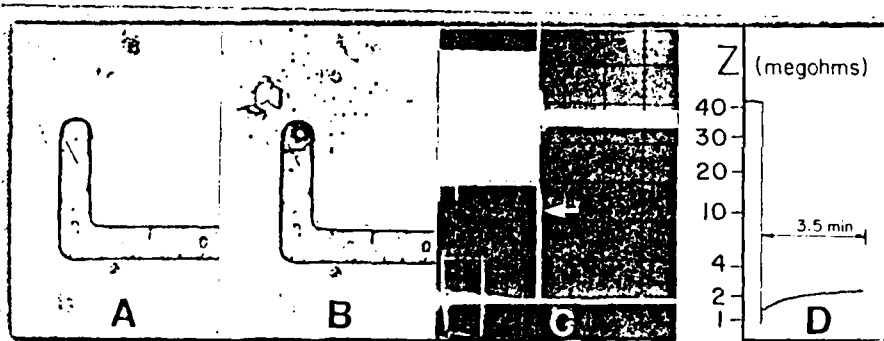
Figs. A4 - A6 From: Gross, G.W. (1979). Simultaneous single unit recording in vitro with a photoetched laser deinsulated gold multi-microelectrode surface. IEEEE Trans. Biomed. Eng. BME-26: 273-279.

A6 Laser-induced electrode deinsulation and concomitant impedance change at 1 kHz. (A) Intact gold conductor 12 μm wide and 2 μm thick covered with a 3-4 μm thick layer of insulation. (B) After single laser shot (337 nm, 1 x 10^10 W/cm²) removed insulation fragments and gold particles can be seen in vicinity of electrode tip. (C) Change in magnitude of 1 kHz sinusoidal signal across electrode at moment of laser exposure (arrow). (D) Similar signal displayed after half-wave rectification on chart recorder. Electrode impedance decreases from 42 MΩ to 1.6 MΩ with one laser shot and rises slowly to 2.2 MΩ within 3.5 min.


Fig. A7 From: Gross, G.W., W.Wen and J. Lin (1985). Transparent indium-tin oxide patterns for extracellular, multisite recording in neuronal cultures. J. Neurosci. Meth. 15: 243-252.

A7 Recording crater impedances as inverse functions of exposed area in μm² for ITO, gold, and ITO that was gold plated in the crater. The linear functions represent normalized impedances of 1130 MΩμm² for ITO and 255 MΩμm² for gold. The high impedance for the ITO-electrolyte interface may be partially a result of further oxidation of the metal oxide during laser deinsulation. Note that gold plating of this interface lowers impedances to those established for gold conductors.
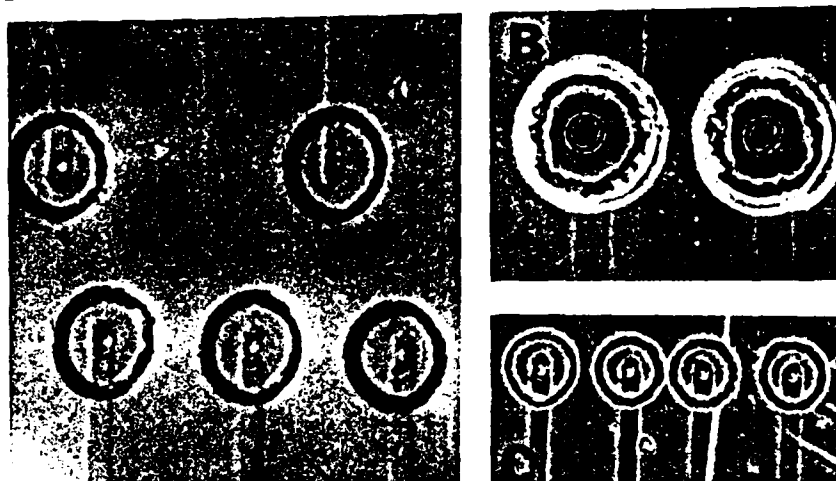
Fig.A8 Typical laser deinsulation patterns in well-cured polysiloxane resin. A: Nomarski micrograph showing shallow, 35 μm diameter craters centered on 10 μm wide conductors. Note the loss of ITO due to the single laser shots at an energy density of 1.5 μJ/μm². B and C: phase contrast micrographs of deinsulation craters formed over 30 μm and 10 μm wide conductors respectively at energy densities ranging from 1.5 to 2.2 μJ/μm².
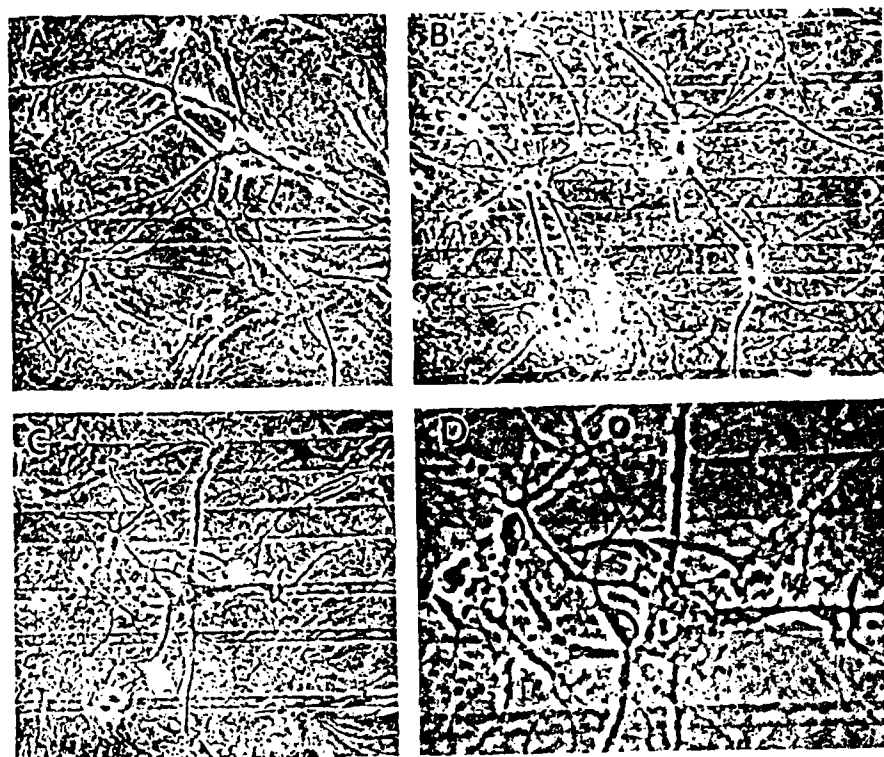
Fig.A9 Representative cultures of spinal neurons growing directly on the ITO, glass surface. ITO appears to be non-toxic since it does not interfere with cell adhesion, growth, and the development of spontaneous electrical activity. A and B: monolayer culture 28 days after seeding. C and D: neuron from a 6-week-old culture at magnifications of 200 and 400 respectively. Note that the 100 nm thick ITO conductors do not interfere with the visualization of neuronal components.

FROM: Gross, G.W., W. Wen and J. Lin (1985). Transparent indium-tin oxide patterns for extracellular, multisite recording in neuronal cultures. J. Neurosci. Meth. 15: 243-252.

A6
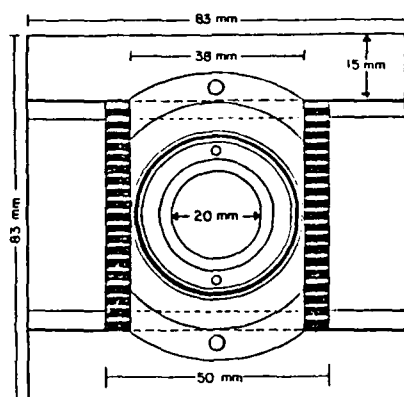
FIG. A10a. Top view of assembled closed chamber and electrode plate holder showing the multielectrode plate with its amplifier contact strips (Zebra strips) to either side and a chamber cover containing the observation window. Medium changes are carried out via the two ports adjacent to the 20mm window.
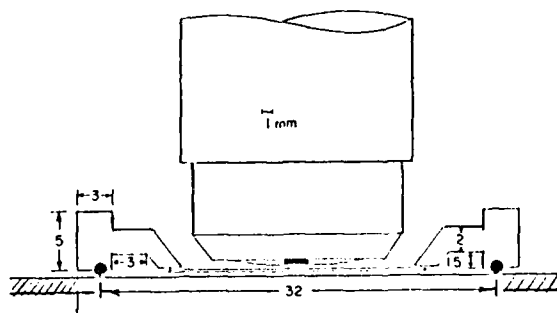
FIG. A10b. Side view of closed chamber containing a 20mm quartz or glass window matched to the objective to be used. This arrangement allows laser cell surgery with Zeiss Ultrafluar x32 and x100 objectives that have working distances of 0.45mm and 0.12mm, respectively.

FROM: Gross, G.W. and M. Hightower (1986) An approach to the determination of network properties in mammalian neuronal monolayer cultures. In: Proceedings of the First IEEE Conference on Synthetic Microstructures in Biological Research, Peckerar, M.C., Shamnia, S.A., and Wyatt, R.J. (eds). Pp. 3 - 21. Washington, D.C.: Naval Research Laboratory.

CLOSED CIRCULATION SYSTEM TO IMPROVE
ELECTROPHYSIOLOGICAL CULTURE STABILITY
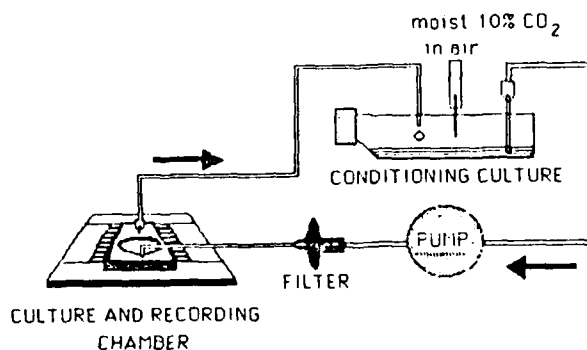
FIG. A11. Schematic drawing showing the recording chamber and closed circulatory system. The recording chamber contains about 300µl of conditioned medium. To maintain pH and osmolarity the medium in the recording chamber is constantly circulated through a 10 ml reservoir of conditioned medium. Moist 10% CO_2 in air is pumped into the reservoir

to maintain pH. An in-line 0.22 µm filter insures sterility in the recording chamber. Pharmacological agents can be added to the reservoir and pumped to the chamber.

A8
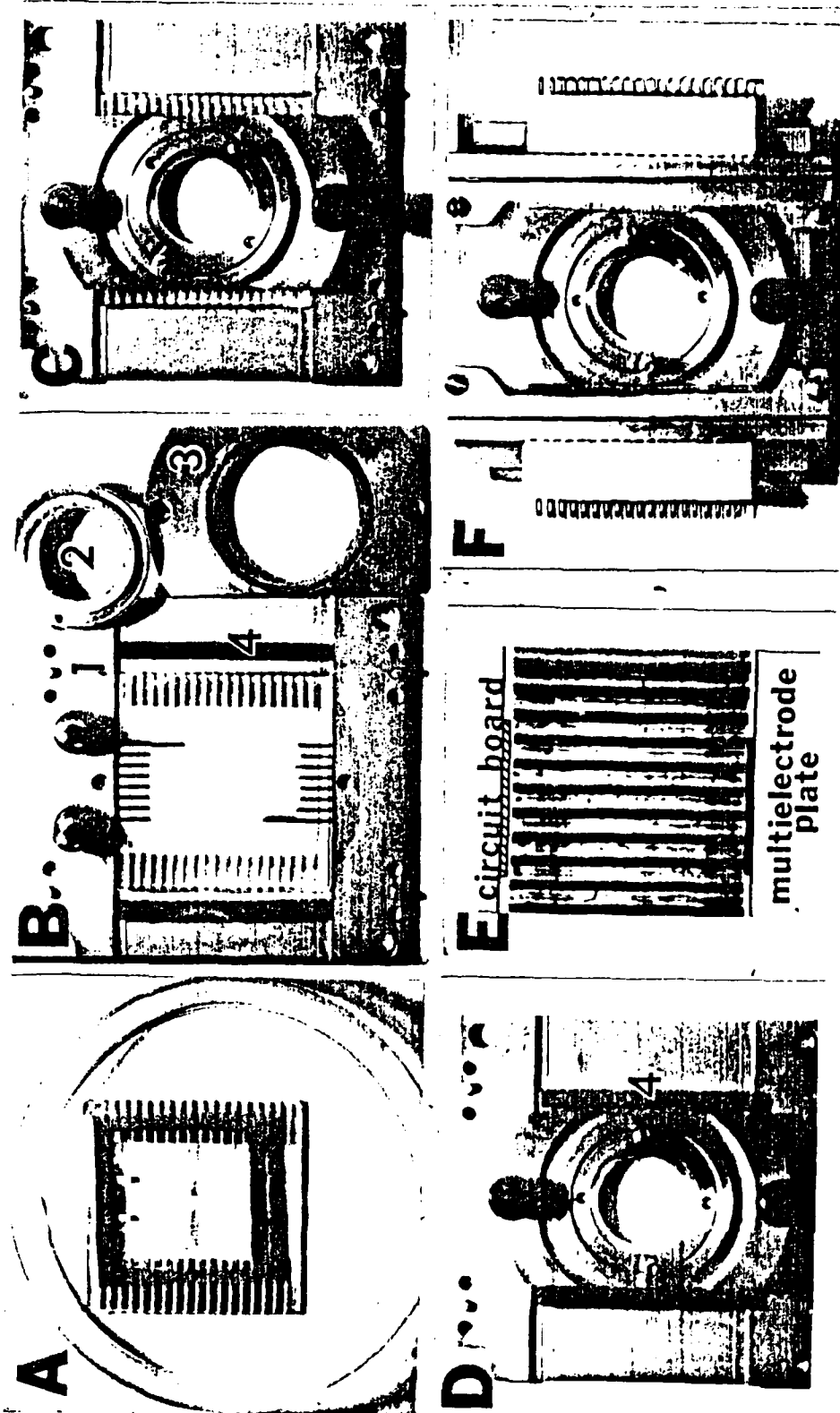
FIG. *A12.* MULTIELECTRODE PLATE AND ASSEMBLY INTO CULTURE CHAMBER THAT ALLOWS SIMULTANEOUS RECORDING, MICROSCOPE OBSERVATIONS, AND LASER CELL SURGERY.

A. Multimicroelectrode plate (MMEP) in culture dish. A 400mm2 culture area is confined by a silicone gasket. Cultures remain in this configuration for 3 to 4 weeks with 50% medium changes every 3 days. B. MMEP in plate holder (1) with quartz microscope port (2), chamber cover (3) and zebra strips (4). C. Assembled chamber; D. Chamber with zebra contact strips in place. E. Closeup of zebra strip showing 70 µm wire loops separated by 100 µm of silicone rubber. The strip makes pressure contact between the MMEP contacts and a circuit board. F. Fully assembled chamber with amplifier connectors.

A9
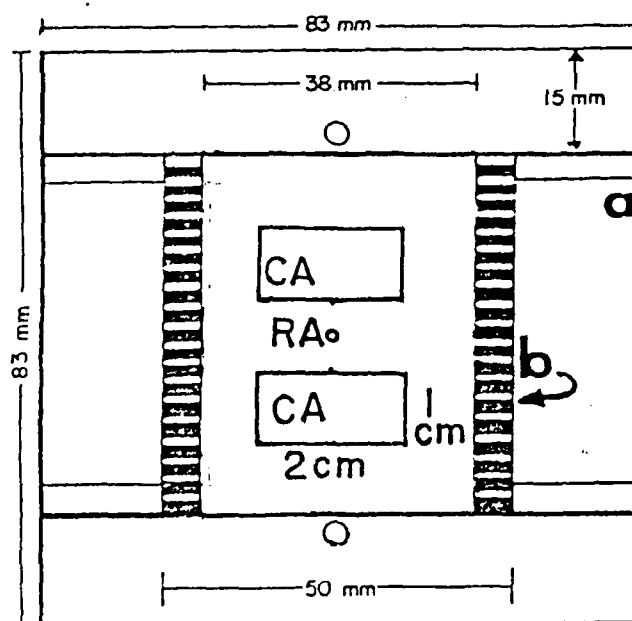
(a)   METAL HOLDER FOR MMEP

(b)   MULTIMICROELECTRODE
       PLATE   (MMEP)

Adhesion Island
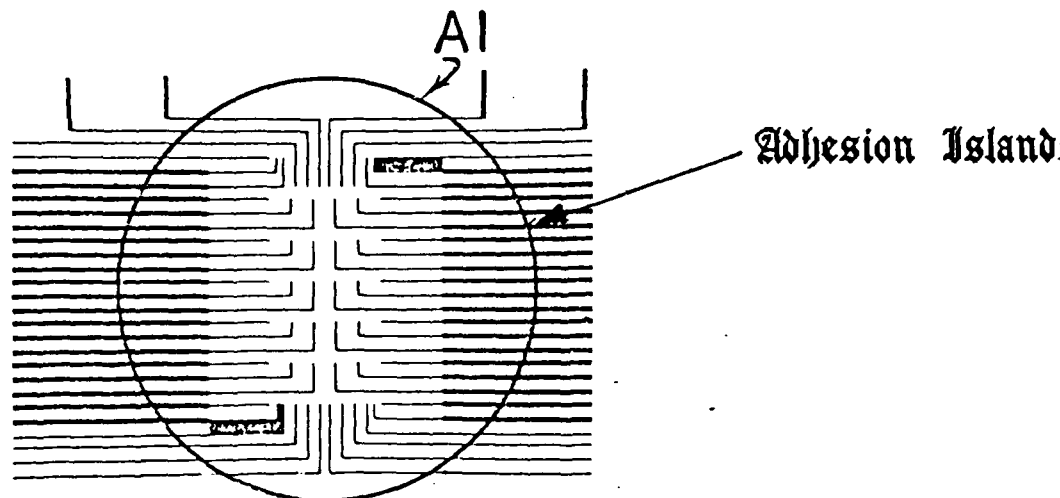
FIG. A13.   CONFINEMENT OF NETWORKS OVER RECORDING AREA VIA SELECTIVE
             ADHESION.

The recording area (RA) is a 0.5mm x 1mm region in the center of the
glass electrode plate where all conductors terminate. Cultures must
be confined to this area to simplify the network analysis. The hydro-
phobic insulation material is flamed through masks to generate specific
adhesion patterns. The pattern shown consists of two "conditioning
areas" to either side of a small (2mm diam) adhesion island (AI) cen-
tered on the recording area. The conditioning areas are necessary for
the proper development of neurons. Neuronal connections between the
three areas do not develop. Medium continuity exists at all times.

A11
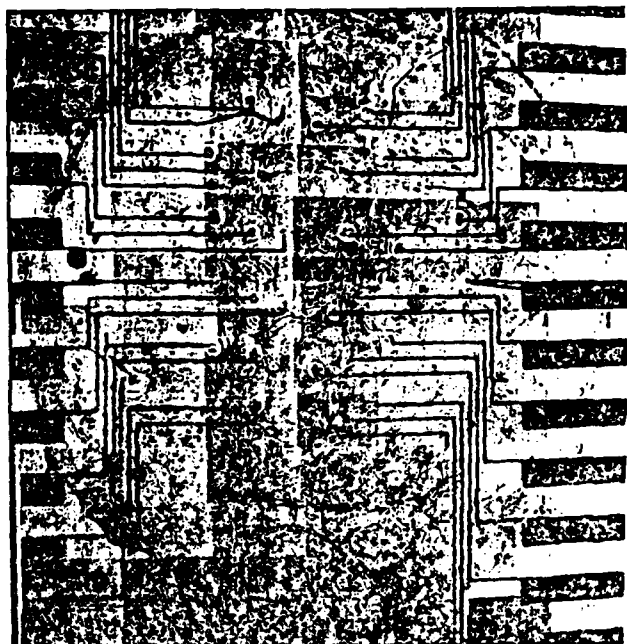
Left: Fig. A14. Monolayer cultures on MMEPs. A 2 mm diameter monolayer culture centered on the recording matrix of a MMEP 1. Adhesion islands are generated on the normally hydrophobic insulation layer with a flaming technique through masks. Laser deinsulation craters are revealed by the halos at the ends of the conductors. Culture density: 400 neurons/mm2.
FROM: Sci. Amer. 256: 62.

Bottom: Fig. A15. Center region of a culture on an ITO MMEP 2. Note that the transparent conductors do not interfere with microscopy. The heavy metal plating in the crater is an artifact of the Bodian histology method due to precipitation of silver and gold onto exposed ITO. All conductors are 10 μm wide.
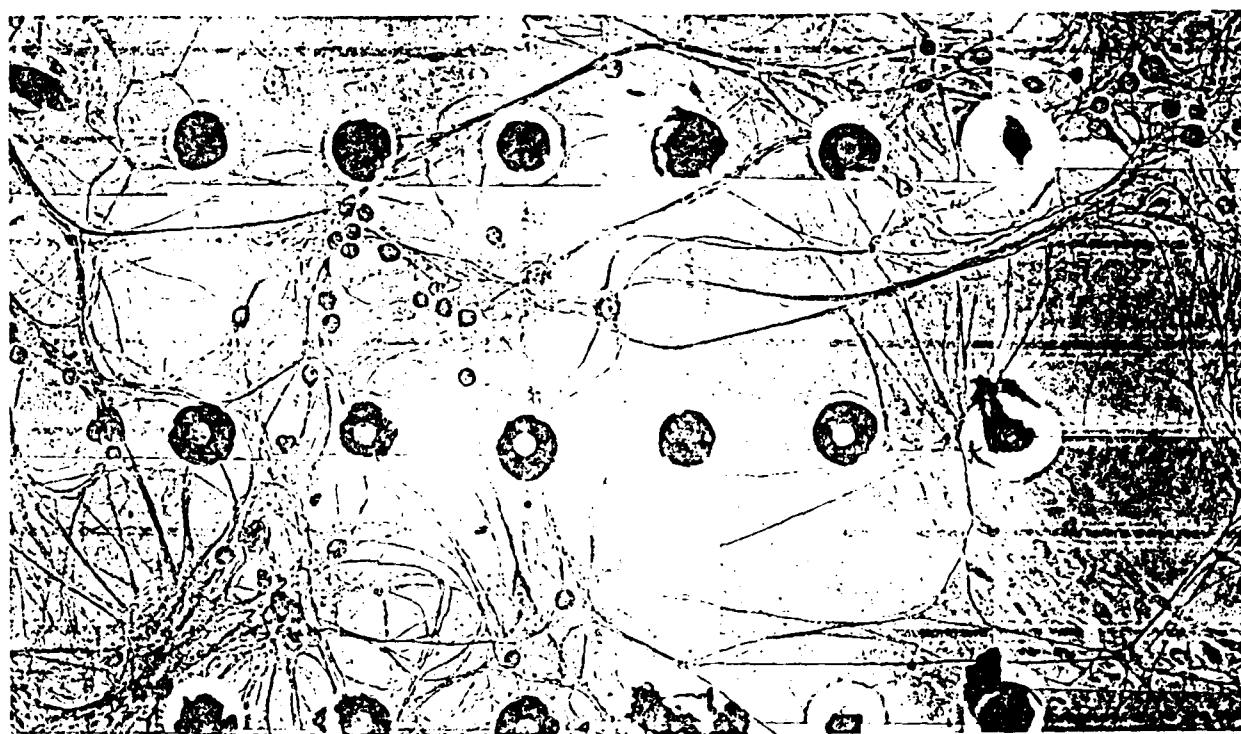
A12
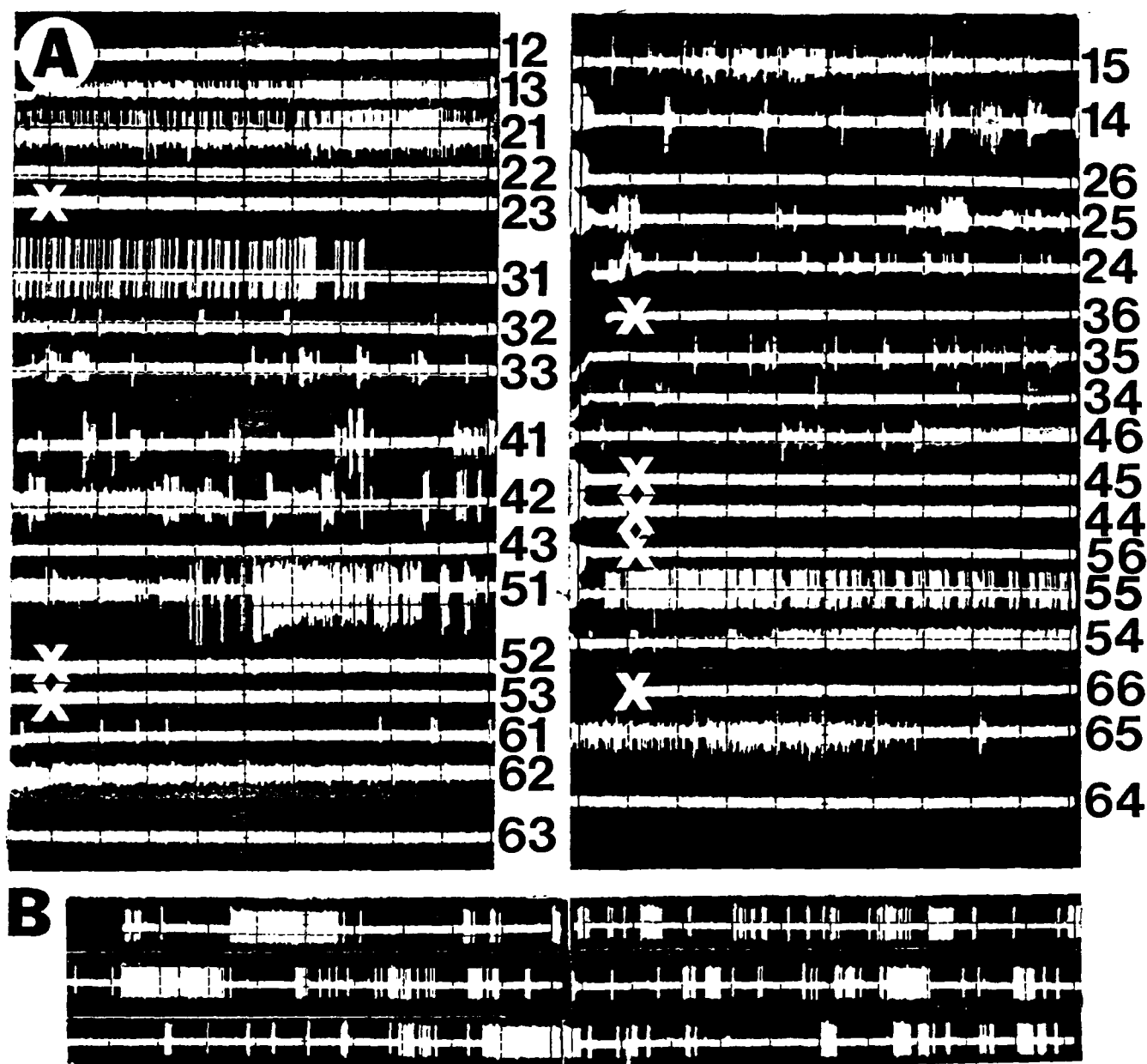
Fig. A16 A. Activity patterns from one culture sampled 34 d after seeding. The data show sequential oscilloscope sweeps (2 sec/div) stored at 40 sec intervals and do not represent simultaneous activity. In this example, bursting is evident on at least 16 of 24 functional electrodes or on 16 of 20 electrodes carrying discernible activity. The *numbers to the right* of each panel represent the row–column identifiers of each electrode. E 63 (sixth row, third column) and E 64 were left insulated to allow a monitoring of shunt-impedance stability. The symbol × denotes electrodes inactive because of conductor discontinuities. *B.* Samples of complex single-unit bursting recorded from E 31 over a 5 min period (sweep, 5.0 sec).

FROM: Droge, M.H., G.W. Gross, M.H. Hightower, and L.E. Czisny (1986) Multielectrode analysis of coordinated, rhythmic bursting in cultured CNS monolayer networks. J. Neurosci. 6: 1583-1592.

A 14
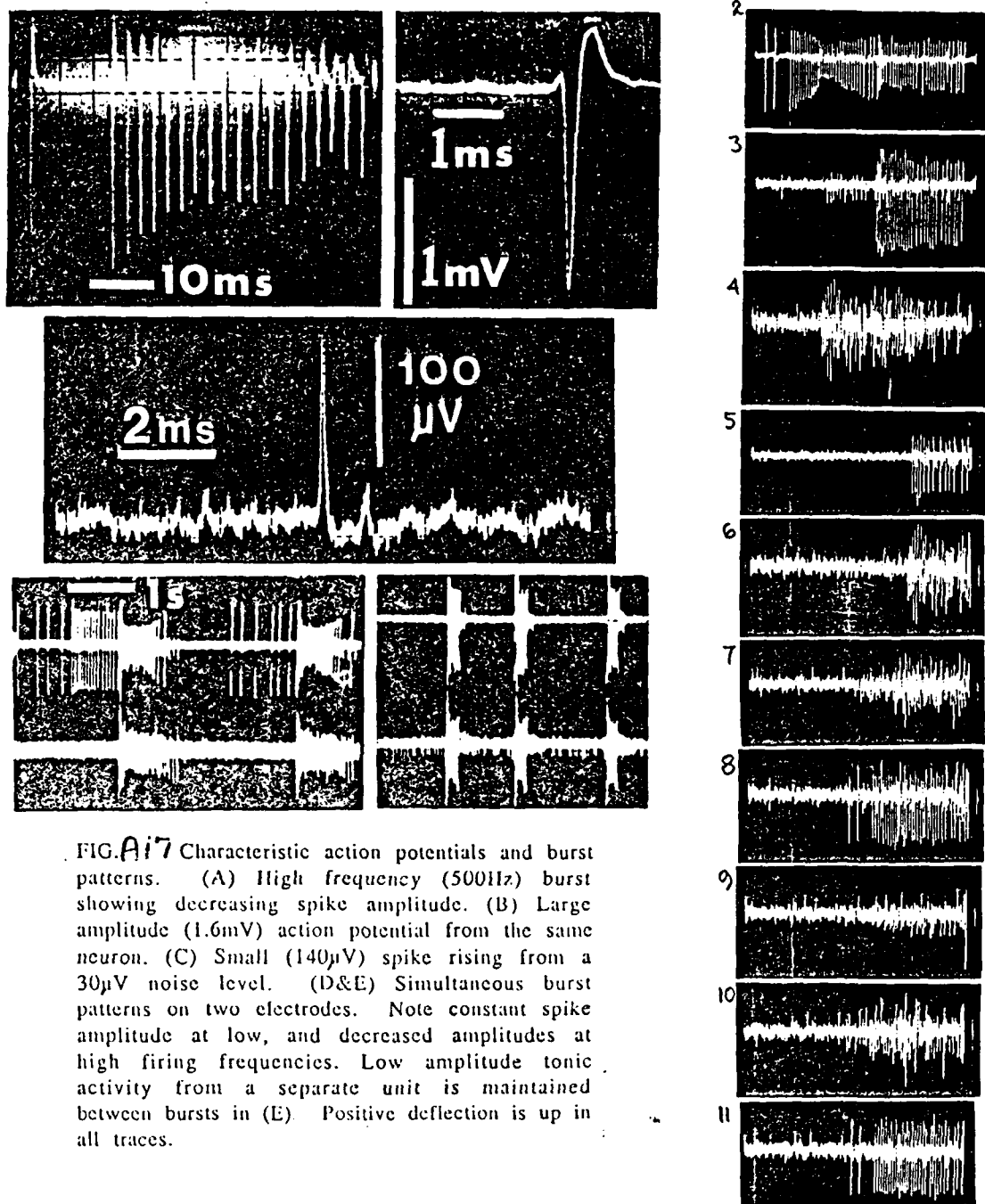
FIG. A17 Characteristic action potentials and burst patterns. (A) High frequency (500Hz) burst showing decreasing spike amplitude. (B) Large amplitude (1.6mV) action potential from the same neuron. (C) Small (140µV) spike rising from a 30µV noise level. (D&E) Simultaneous burst patterns on two electrodes. Note constant spike amplitude at low, and decreased amplitudes at high firing frequencies. Low amplitude tonic activity from a separate unit is maintained between bursts in (E). Positive deflection is up in all traces.

Gross, G.W. and M.L. Hightower (1987) Multielectrode investigations of network properties in Neural monolayer cultures. In: Biomedical Engineering, Recent Advances, (R.C. Eberhardt, Ed.), McGregor and Werner, Washington; in press.

10 CHANNEL SIMULTANEOUS BURST DATA
Analog storage scope;
sweep: 20 ms
noise: 40 µV
(unpublished)

A15
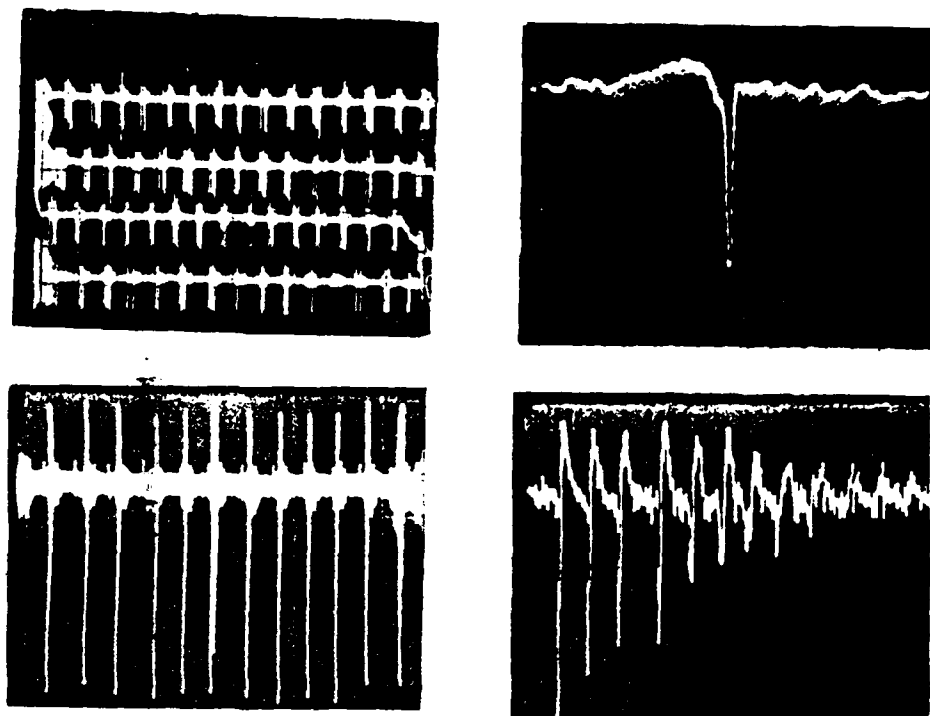
Fig. A18 Oscilloscope tracings from olfactory co-cultures (dissociated olfactory bulb cells cultured with explants of olfactory neuroepithelium) grown on MMEPs. Upper left: sustained, rhythmic burst activity; each line contains ten seconds of data and the average amplitude of each burst is 200μV (peak to peak). Upper right: single unit action potential; the entire tracing represents 20 msec and the amplitude of the action potential is about 1mV (p/p). Lower left: single unit action potential showing amplitude decay; the entire tracing represents 10 msec and the amplitude of the largest action potentials are about 800μV (p/p). Lower right: expanded tracing of similar tracings to the right illustrating amplitude decay and waveform alteration; the entire tracing represents 50 msec and the amplitude of the largest action potentials are about 800μV (p/p).

A16

A 17

MMEP Monitoring
Hardware

Auditory Monitor   LED Display

Neural Culture

120 Foot Pipe

Patch Pannel

FET Stage Amplifiers

Amplifiers

Integrators

Masscomp 5700   Graphics Display   Stripchart Recorder   Oscilloscope

Fig. A 19   Present data acquisition and processing setup. The first stage amplifiers reside on the
microscope stage to either side to the MMEP. Second stage amps are connected to a patch panel, to an LED
display, to an auditory monitor and to the Masscomp 5700 computer. Integrators, an 8 channel strip
chart, and oscilloscopes are serviced by the patch panel. The LED display represents the physical
layout of the electrodes on the MMEP and displays activity in a three color sequence (green - yellow -
red) depending on spike intensity. The auditory monitor is fed by the LED circuit and presents activity
on each electrode as a different carrier frequency. Teh later two analog devices are very useful for
determining which electrodes are active and also for the recogniton of patterns.

A-18

Fig A 20

MASSCOMP 5700: Current Configuration
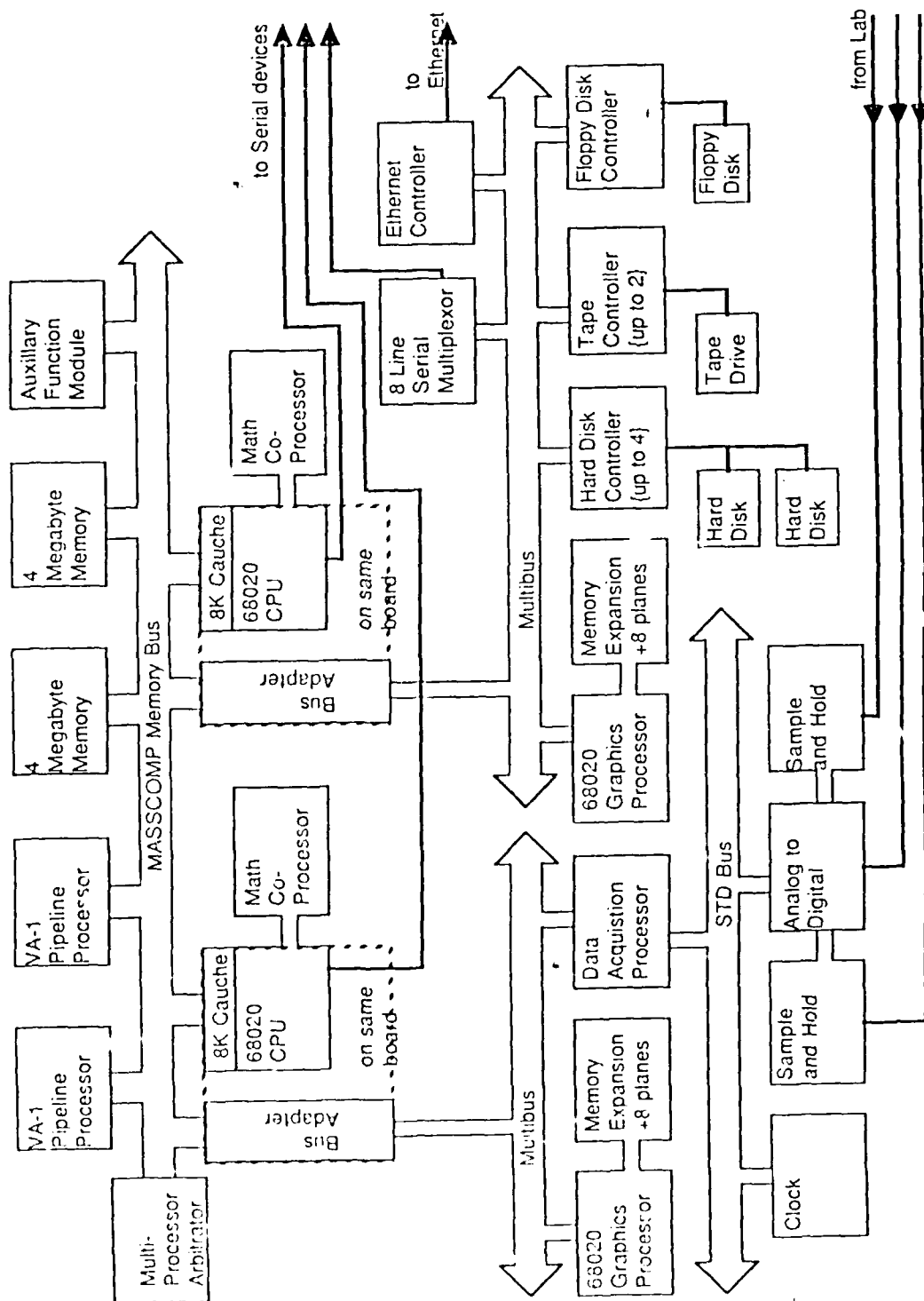
Poj A 19

PRESENT COMPUTER HARDWARE

1.0 SCOPE

This document outlines the computational facilities available at the Center for Network Neuroscience. The principle system is a Masscomp 5700 parallel processing computer. The Masscomp 5700 is intended for algorithm development, statistical manipulation, and real time experiment monitoring. The center also has a MicroVax for experiment control and a network of Macintosh Computers for document preparation.

2.0 MASSCOMP 5700

2.1 System Definition

The Masscomp 5700 is a computer mainframe capable of clustering four different types of specialized processors. The mainframe is developed on the Motorola 68000 family of computer processors. The processors include a standard CPU, a Data Acquisition and Control Processor (DACP), a Pipeline Processor, and a Graphics Processor. The current system configuration employs a single DACP and two of each of the other processors.

The system provides access to two industry standard busses for peripherals as well as a high-speed main bus. Multibus provides access to standard computer peripherals such as disk drives and tapes. STD bus is used for experimental instrumentation.

2.1.1 Functional Description

2 1.1.1 The Standard CPU

Masscomp has two different modules that can be used for the standard CPU. The Center has two of the 68020 modules. The 68020 module contains a 68020 CPU, 68881 Math Coprocessor, an 8K Cache area, and a Multibus Adapter. The math expansion module, the lightning floating point module, expands the throughput of the math co-processor on scientific functions. Both of the 68020 modules are equipped with the lighting boards. Each processor is capable of about 3 Mflops a second.

2.1.1.2 The Data Acquisition and Control Processor

The DACP is an 8 MHz bit-slice processor that is intended for realtime operations. The DACP is located on multibus and provides an adapter to STD bus. The DACP controls service interrupts from the STD bus modules and load blocks of read data into main memory. The center currently has four STD bus modules, a clock, a 1 MHz A/D, and two sample and hold modules.

2.1.1.3 The Pipeline Processor

The math pipeline processor uses a 7.1 MHz adder and multiplier pipe to provide a performance of 14.2 Mflops per second. The system has an instruction queue for DMA operations as well as for math operation. DMA can be performed simultaneously with math operations in different sections of the 128KB of memory.

A - 20

2.1.1.4 Graphics Processor

The graphics processor accepts high level graphic commands for an Aurora
Display. The Aurora is an 1150 x 910 pixel display with 4096 displayable colors out
of a 16 million color palette. The graphics processor also controls the I/O from the
Aurora keyboard and mouse.

2.1.2 Peripherals

2.1.2.1 Memory

The system is configured with 8 MB of main memory and 6 MB of graphics
memory located on multibus.

2.1.2.2 Mass Storage

The system contains two Fujitsu Eagle disk drives as the principle mass storage
device. Each Eagle has 387 MB of disk storage. The system is also equipped with a
1/2" tape drive for doing backups and a 5 1/4" floppy disk drive for system
configuration and software updates.

2.1.2.3 I/O

The system has 14 RS-232 Serial Ports and an Ethernet connection. The system
uses two of the RS-232 ports for interfacing to the NTSU broadband network for
terminal access and another RS-232 connection to a 1200 baud modem. The system
is also connected to two dot-matrix printers, and a VT100 that serves as system
monitor.

2.2 Performance

Each standard CPU has a benchmark of 3300 Kwhetstones. Each pipeline processor is
capable of 14.2 Mflops sustained throughput. The combined throughput of two standard
CPU modules and two pipeline processors is estimated at 35 Mflops.

2.3 Physical Attributes

The system is housed in two cabinets. The primary cabinet houses the 30 slot frame for
Masscomp bus and multibus boards, one Eagle hard disk, and two 9 slot STD bus frames.
The second cabinet contains a second Eagle and a tape drive.

2.4 Maintenance and Support

The system is maintained on a service contract that provides for replacement of defective
hardware, as well as software support and routine system maintenance.

A-21

3.0 MICROVAX

3.1 System Definition

The MicroVax was purchased to do processor control of lab equipment. The control process will be done through a CAMAC interface.

3.1.1 Functional Description

The system is a single board version of the Vax mainframe produced by Digital Equipment Corporation. The system is equivalent to a 11/785 with a math co-processor with slightly slower bus hardware.

The system is also equipped with a graphics processor with a display of 1024 x 1024 pixels. Each pixel can be assigned one of 16 colors selected from a 4096 color palette.

3.1.2 Peripherals

3.1.2.1 Memory

The MicroVax is configured with 3 MB of memory.

3.1.2.2 Mass Storage

The MicroVax has a 70 MB Winchester disk drive with a T50 streaming tape backup.

3.1.2.3 I/O

The MicroVax has two RS-232 ports and an Ethernet connection.

3.2 Performance

The MicroVax is benchmarked at 1000 kilowhetstones.

3.3 Physical Attributes

The MicroVax fits into a standard rack mount cabinet using 5" of space.

3.4 Maintenance and Support

The system is being serviced as needed, with billings for time and materials. Software support is being provided through campus computer center.

A-22

4.0 MACHINTOSH NETWORK

4.1 System Definition

The System is a network of six workstations connected to a laser printer. The system provides the Center with document formating from personal computers.

4.1.1 Functional Description

4.1.1.1 Network Description

The Network uses Applenet to connect the workstations and laser printer together. Applenet is a broadband network similar to ethernet with a 1/4 Mhz bandwidth.

4.1.1.2 Workstation Description

The center has five Mac+ workstations and a Mac 2 workstation. The Mac+ is a 1MB system with a black and white screen. The Mac 2 is a color system with 16 color pixels and a 40 MB Winchester hard disk.

4.1.1.3 Printers

The main printer is an Apple Laserwriter printer. The Laserwriter is capable of printing at the rate of 8 pages a minute. Two workstations have 2 dot matrix printers attached for rough drafts.

4.2 Maintenance and Support

Hardware failures are fixed as needed. Software support is supplied by the vendor.

A-23

Data Acquisition

16 channels of analog data can be sampled at 24,000Hz each. Signals are synchronized through a pair of Sample&Hold boards. The analog data is then converted to digital information through a central 12-bit A/D. The combined sampling rate is 1.728 Mbytes per second.

Matched AC Filter and DC Offset Adjustment

AC line related noise is removed by subtracting a template of the line noise from the signal. The template is generated by an exponentially weighted average, and DC offsets from the A/D are represented in this average. The entire operation requires an arithmatic shift, an increment, and two adds.

Rectification and Compression

Information on each channel is compressed by recording the maximum absolute value from the 8 samples contained within each 1/3ms bin. This operation results in a 8:1 compression.

Activity Detection by an Adaptive Threshold

The compressed data are used to estimate the noise standard deviation. A signal threshold is established at 4 noise standard deviations. Activity in any 1/3ms bin above this threshold will be detected by this system. 15 values are packed into a word for an additional 15:1 compression.

Integration

Neurophysiologists have used integrated chart recordings to study burst patterns. The integration is performed by a resistor/capacitor network. Such an R/C network is modeled to mimic integrated activity by the computer software using empirically derived leakage constants.

Burst Detection

Bursting is detected when the integrated activity exceeds threshold. Burst amplitude, duration, period, area, and other parameters are computed and stored for statistical analysis as well as a envelope shape classification.

Pg A 24

Fig. A20 Emerging realtime display program for 36 electrodes showing spike data (top) and integrated data (bottom). All electrodes are displayed in the lower right of the screen in a physical arrangement that mimics the layout of the electrodes under the microscope (MEP 1). The center of the square electrode selection buttons will show different colors as a function of burst intensity (not completed). The square collar has a color code that corresponds to the trace position on the screen. Sweep speeds and amplitudes are selected at the bottom of the screen. A total of 6 channels will eventually appear on the screen. Statistical parameters such as average burst duration, burst period, burst area, and burst type (as well as various histograms) will also be selectable from the panel above the electrode matrix. Most parameters can be plotted as a function of time and will provide a continuous 6 hour record of these parameters.

Because of our constant exposure to multichannel data, we have a high probability of developing realistic, effective, operator-friendly programs that might be used for many different multichannel problems. It is conceivable that EEG data could be selected in the same way from a brain/electrode schematic that replaces the 6x6 electrode matrix.

A25

1.0 s

1. Sample from analog-to-digital converter.

2. Filtered for A/C line related noise.

3. Rectification and Compression to 3000 hz.

4. Binary channel indicating activity.

5. R/C integration of the binary Channel.
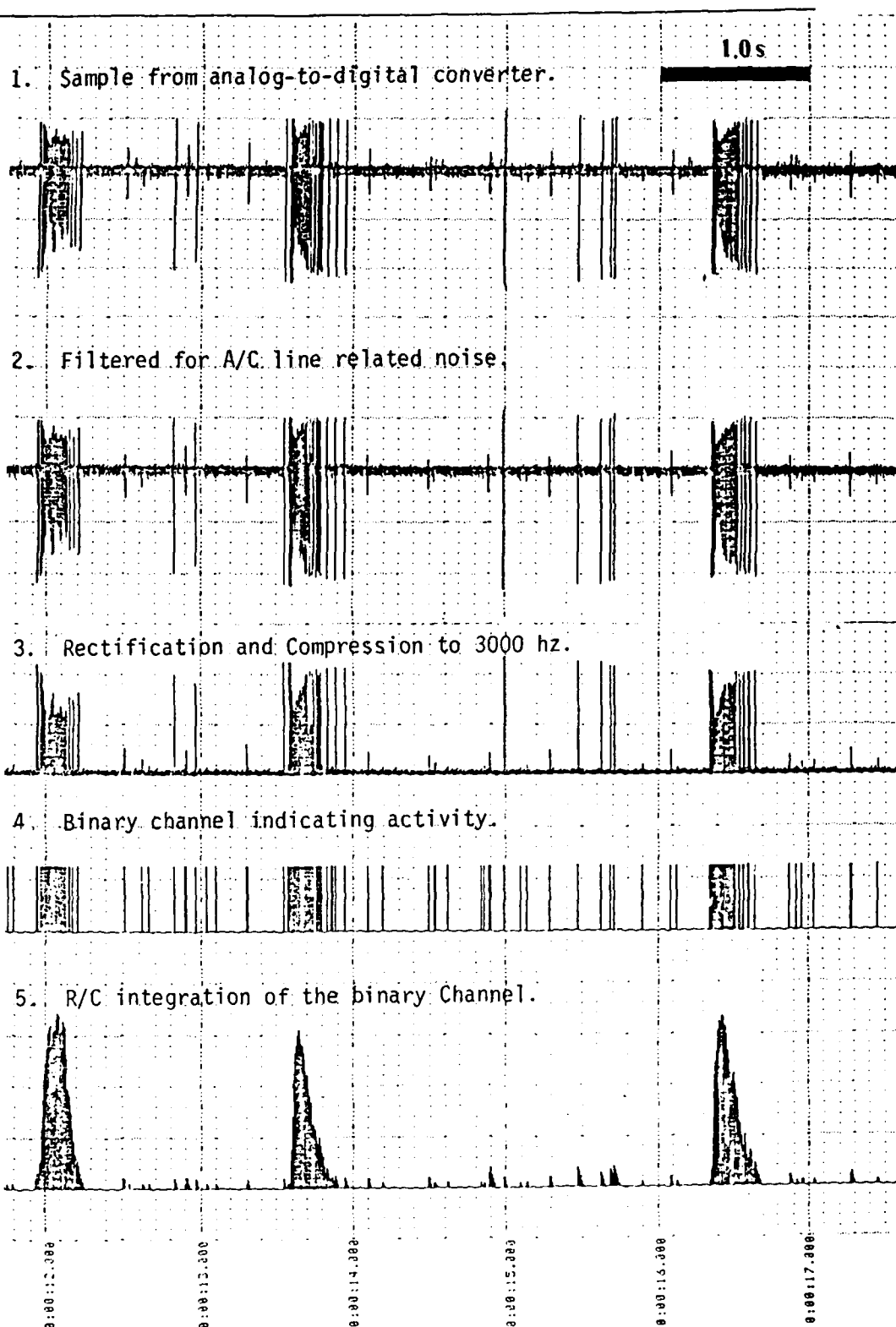
Data processing example. Sequential modification of the data sample results finally in R/C integration of the binary channel.

Pg A26

0.1 s

1. Sample from analog-to-digital converter.

2. Filtered for A/C line related noise.

3. Rectification and Compression to 3000 hz.

4. Binary channel indicating activity.

5. R/C integration of the binary Channel.
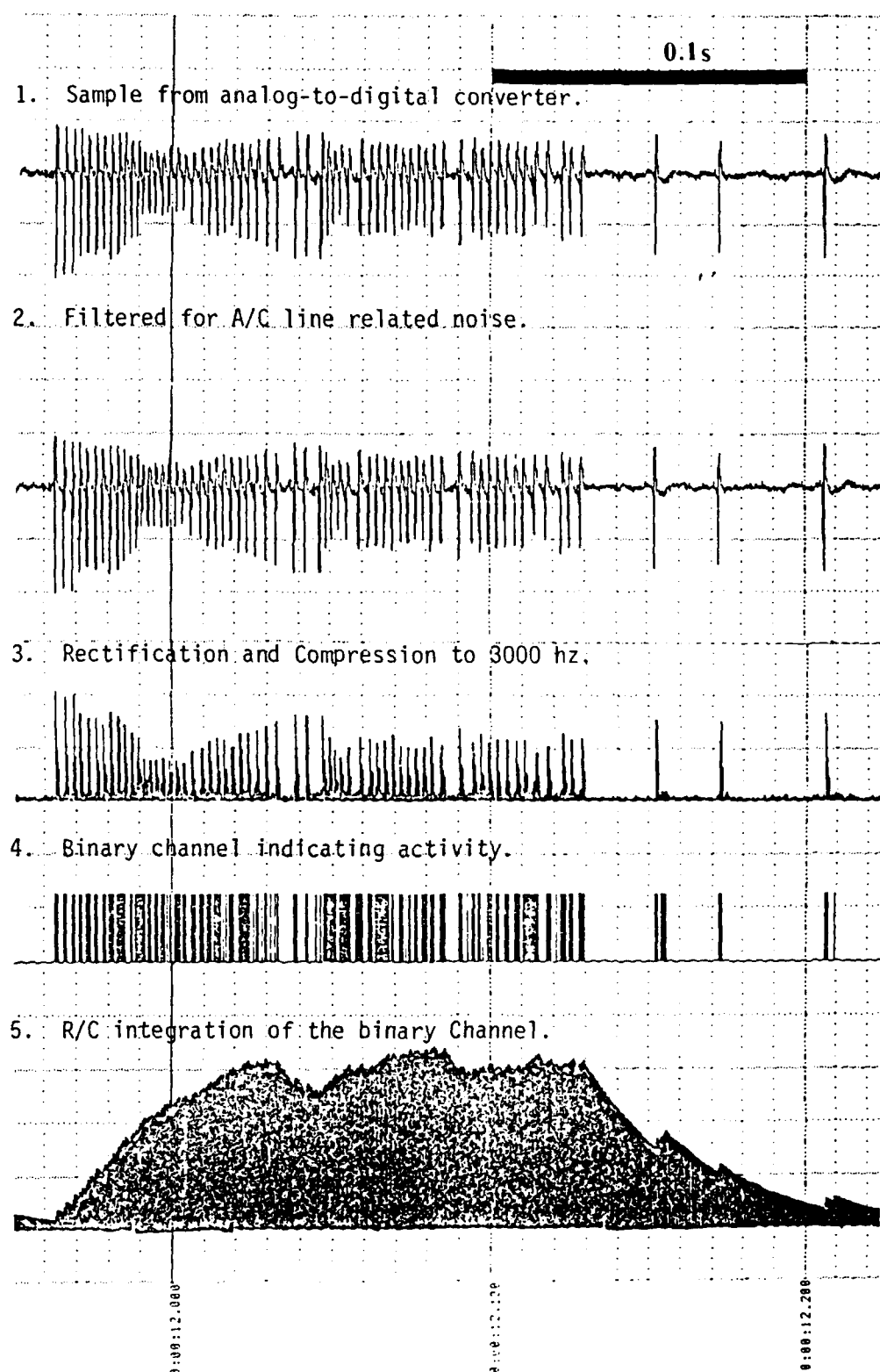
A faster sweep of the data sample from the preceding page, showing the recording of waveforms of individual spikes, with the resulting R/C integration of the binary channel.

Pg A27
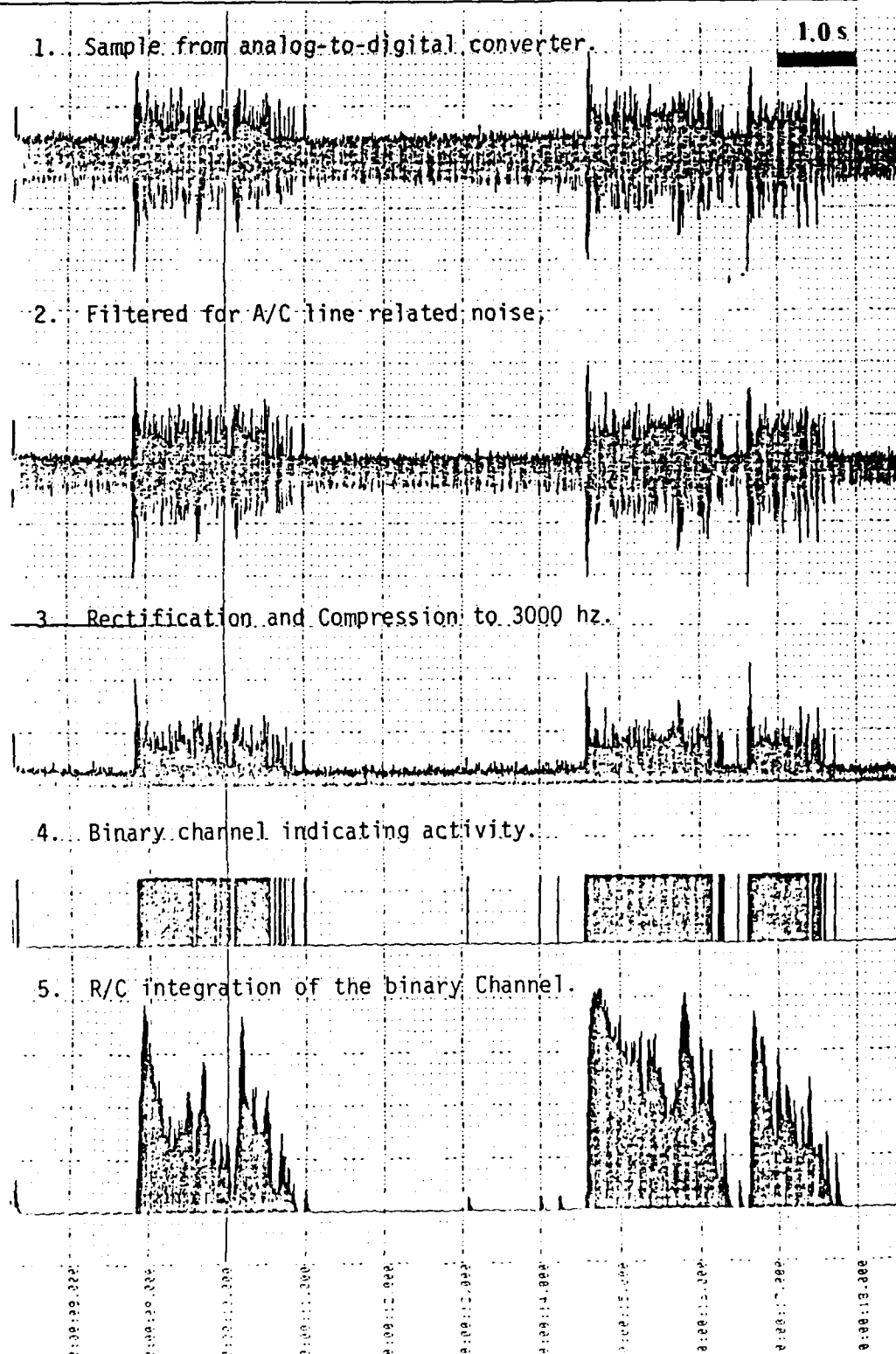
A second data processing example, exhibiting a higher level of A/C noise. After
filtering for line noise, the R/C integration of the binary channel produces an
electrically "clean" signal.

Pg A28

1. Sample from analog-to-digital converter.

0.1 s

2. Filtered for A/C line related noise.

3. Rectification and Compression to 3000 hz.

4. Binary channel indicating activity.

5. R/C integration of the binary Channel.
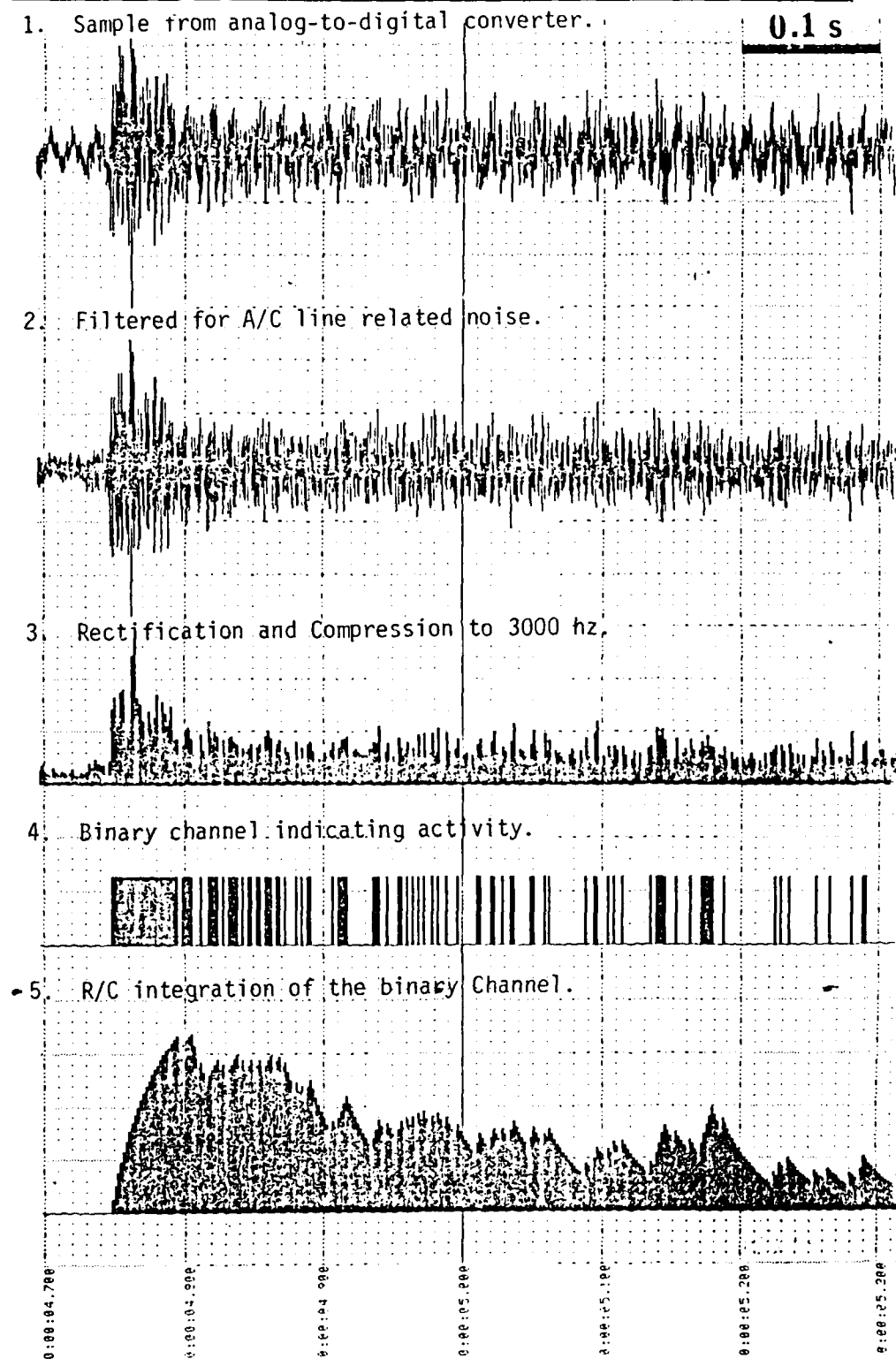
A faster sweep of the data sample from the preceding page, better illustrating the effects of high line noise and its correction by filtering.

Pg A29

APPENDIX B

SOFTWARE LISTINGS

```
    1            PROGRAM DTEST
    2 C
    3 C        PROGRAM TO TEST THE "GIBBS" SUBROUTINE (RLD 12/02/87)
    4 C
    5            DIMENSION X(8,2048),Y(8,2048), TILEX(256), TILEY(256)
    6            INTEGER*2  HISTO(0:255,0:255), INDEX(2048)
    7 C
    8              CHARACTER*8    R4NAMES, I4NAMES, L1NAMES
    9              REAL*4         R4VALUE
   10              INTEGER*4      I4VALUE
   11              LOGICAL*1      L1VALUE,QUIT,PLOT
   12 C
   13            COMMON /NLIST/ R4NAMES(21),I4NAMES(21),L1NAMES(7),
   14       1                   R4VALUE(21),I4VALUE(21),L1VALUE(7)
   15 C
   16            EQUIVALENCE (SEED,R4VALUE(1)),(A,R4VALUE(2)),(B,R4VALUE(3)),
   17       1            (GOTIME,R4VALUE(4))
   18            EQUIVALENCE (NX,I4VALUE(1)),(NY,I4VALUE(2)),(LEXP,I4VALUE(3)),
   19       2     (INSTR,I4VALUE(4)),(IOUNIT,I4VALUE(5)),(IC,I4VALUE(6)),
   20       2     (IPR,I4VALUE(7)),(IDU,I4VALUE(8)),(IFUN,I4VALUE(9)),
   21       2     (LDBUG,I4VALUE(10))
   22            EQUIVALENCE (QUIT,L1VALUE(1))
   23 C
   24 C  .
   25            NX = 8
   26            NY = 8
   27            IC = 8
   28            LEN = 255
   29            SEED  = 2.468E+12
   30            A = 0.
   31            B = 1.
   32            LEXP = IC * 2**MAX(NX,NY)
   33            INSTR = 1
   34            QUIT=.FALSE.
   35            IOUNIT = 6
   36            IDU   = 0
   37            IPR   = 0
   38            IFUN  = 0
   39            LDBUG  = 0
   40 C
   41            R4NAMES(1) = 'SEED'
   42            R4NAMES(2) = 'A'
   43            R4NAMES(3) = 'B'
   44            R4NAMES(4) = 'GOTIME'
   45 C
   46            I4NAMES(1) = 'NX'
   47            I4NAMES(2) = 'NY'
   48            I4NAMES(3) = 'LEXP'
   49            I4NAMES(4) = 'INSTR'
   50            I4NAMES(5) = 'INPTUNIT'
   51            I4NAMES(6) = 'IC'
   52            I4NAMES(7) = 'PRNTUNIT'
   53            I4NAMES(8) = 'DATAUNIT'
   54            I4NAMES(9) = 'IFUN'
   55            I4NAMES(10)= 'LDBUG'
   56 C
   57            L1NAMES(1) = 'QUIT'
   58 C
   59            NL1 = 4
   60            NL2 = 10
```

```
 61              NL3 = 1
 62              TIME=0.
 63 C
 64 10          CONTINUE
 65 C
 66              CALL NAMELIST(IOUNIT,NL1,NL2,NL3)
 67              IF(QUIT) STOP
 68              ASEED = SEED
 69              NHX = 2**NX-1
 70              NHY = 2**NY-1
 71              NTRIALS = 0
 72
 73              IF(IDU.GT.0 .AND. GOTIME.LT.TIME) REWIND(IDU)
 74              TIME = GOTIME
 75 C
 76              IF(INSTR.EQ.2) THEN
 77                 WRITE(9,11) (R4NAMES(I),R4VALUE(I),I=1,NL1)
 78                 WRITE(9,12) (I4NAMES(I),I4VALUE(I),I=1,NL2)
 79 11              FORMAT((4(A8,'= ',F8.4,1X)))
 80 12              FORMAT((4(A8,'= ',I8,1X)))
 81              END IF
 82 C
 83 C        GENERATE OR READ THE DATA
 84    .    PRINT *,' DTEST:  Generating random data for X and Y.'
 85 C
 86              DO 20 ITRIAL = 1,LEXP
 87
 88          IF(IDU.LE.0) THEN
 89              CALL QRX(X(1,ITRIAL),NX,Y(1,ITRIAL),NY,A,B,ASEED,IFUN)
 90          ELSE
 91              STOP 'DTEST: Real data initialization not available.'
 92 C            CALL RDATA(X,N,TIME,IDU)
 93          END IF
 94
 95 20      CONTINUE
 96
 97          IF(LDBUG.GE.3) WRITE(6,902) ((X(I,J),I=1,NX),J=1,LEXP)
 98 902     FORMAT(8(1X,F8.5))
 99
100 C        GENERATE THE X AND Y EVENT-SPACE TILINGS
101
102          PRINT *,' DTEST:  Generating the X event-space tilings.'
103          CALL UNIVENT(X,NX,TILEX,INDEX,IC,LDBUG)
104          PRINT *,' DTEST:  Generating the Y event-space tilings.'
105          CALL UNIVENT(Y,NY,TILEY,INDEX,IC,LDBUG)
106
107 C        COMPUTE THE NORMALIZED STRUCTURE INDEX.
108          PRINT *,' DTEST:  Computing the normalized structure index.'
109
110          DO 100 ITRIAL = 1,LEXP
111
112          CALL GIBBS(X(1,ITRIAL),NX,TILEX, Y(1,ITRIAL),NY,TILEY, NTRIALS,
113        +           HISTO, NHX,NHY, INSTR, HX,HY,H,G,LDBUG)
114
115              IF(INSTR.EQ.2) THEN
116                  WRITE(9,101) H,HX,HY,G
117 101              FORMAT(4(F12.5,1X))
118              END IF
119
120 100     CONTINUE
```

```
121 C
122           NTR = NTRIALS
123           CALL GIBBS(X,NX,TILEX, Y,NY,TILEY, NTR,
124        +             HISTO, NHX,NHY, O, HX,HY,H,G,LDBUG)
125 C
126           IF(IPR.NE.6) WRITE(6,901) NTRIALS,SEED,NX,NY,H,HX,HY,G
127 C
128           IF(IPR.GT.0) WRITE(IPR,901) NTRIALS,SEED,NX,NY,H,HX,HY,G
129 901       FORMAT(' DMORPH EXPERIMENT:  # TRIALS = ',I5,', SEED = ',
130        1  F10.7,',  X-DIM = ',I2,',  Y-DIM = ',I2,
131        2  /8X,'WHOLE ENTROPY = ',F10.7,',  X ENTROPY = ',F10.7,
132        3  /8X,'Y ENTROPY = ',F10.7,',  DMORPH       = ',F10.7,/)
133 C
134           GO TO 10
135           END




NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0




136
137           SUBROUTINE QRX(X,NX,Y,NY,A,B,SEED,IFUN)
138           DIMENSION X(NX),Y(NY)
139 C
140               DO 100 I = 1,NX
141                   X(I) = A + (B-A)*URANF(SEED)
142 100       CONTINUE
143 C
144               DO 200 I = 1,NY
145                   Y(I) = A + (B-A)*URANF(SEED)
146 200       CONTINUE
147
148           GOTO(101,102,103,104) IFUN
149           GOTO 1000
150 C
151 101       CONTINUE
152 C         X(4) = Y(2) + X(4)
153           GOTO 1000
154 C
155 102       CONTINUE
156           X(4) = (Y(2)+X(4))/2.
157           GOTO 1000
158
159 103       CONTINUE
160           DO 1031 I=1,NX
161 1031      X(I) = Y(I)
162           GOTO 1000
163
164 104       CONTINUE
165           X(2) = X(1)
166
167 1000      RETURN
168           END
```

NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0


```
169
170         SUBROUTINE RDATA(X,N,T,IDU,ITRIAL)
171 C
172 C     This subroutine reads sample data from the file FORTn, where
173 C     n = IDU.  It skips all records with time-tags less than T,
174 C     reads all records with time-tags equal to the time-value first
175 C     encountered which is greater than or equal to T, and puts the
176 C     next value of the time-tag into the T variable before returning.
177 C
178 C     If you want to interpolate or extend the data between times
179 C     existing on the file, you must do that external to this subroutine.
180 C
181         DIMENSION X(N),M(4),R(4)
182
183         IF(ITRIAL.GT.1) GOTO 20
184
185 10      CONTINUE
186
187         READ(IDU,901,END=1200) TT,NX,GX,(M(I),R(I),I=1,4)
188 901     FORMAT(F9.3,1X,I1,1X,F8.3,1X,4(I3,1X,F10.3,1X))
189
190 20      CONTINUE
191         IF(TT.LT.T) GOTO 10
192
193         DO 100,I=1,4
194 100     IF(M(I).GT.0 .AND. M(I).LE.N) X(M(I))=R(I)
195
196         TT1 = TT
197         READ(IDU,901,END=1100) TT,NX,GX,(M(I),R(I),I=1,4)
198         IF(TT.EQ.TT1) GOTO 20
199         T = TT
200
201 1100    CONTINUE
202         RETURN
203 1200    CONTINUE
204         PRINT *,' NO DATA EXISTS ON INPUT DATA UNIT ',IDU
205         PRINT *,' BEYOND THE REQUESTED TIME   T = ',T
206         STOP
207         END
```


NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0

NUMBER OF WARNINGS IN COMPILATION : 0
NUMBER OF ERRORS   IN COMPILATION : 0

```
     1          SUBROUTINE UNIVENT(X, NX, THRESH, INDEX, IC, LDBUG)
     2
     3 C*********************************************************************
     4 C          This routine determines the SUM (i=1,K) 2**(I-1) threshholds
     5 C          which will divide the k-dimensional space into regions that
     6 C          will have an equal amounts of counts when the sample is
     7 C          drawn from the same underlying distribution which generated
     8 C          the c*2**K vectors used by this program to set the boundaries.
     9 C          If IC=4, K=8 and the data is found on IUNIT = 1 then the num.
    10 C          of threshholds which need to be found are 1+2+4+8+16+32+64+128
    11 C          or 255 based on the 512 data vectors.
    12 C*********************************************************************
    13
    14 C*********************************************************************
    15 C
    16 C     IC is the integer multiple of the min. # of samples(2**NX).
    17 C     NX is the sample vector dimension.
    18 C     NSAMPLE = IC * 2 ** NX, is the number of samples.
    19 C     NTHRESH = 2 ** NX - 1, is the total number of thresholds.
    20 C     DATA(I,J), i=1,NX ), J=1,NSAMPLE ) is the sampled data.
    21 C     THRESH is the array in which the thresholds are stored.
    22 C     LTHR   is the length of THRESH and must = -1+2**NX
    23 C     INDEX  is a workspace integer array of length NSAMPLE.
    24 C  .
    25 C     The calling sequence for UNIVENT is as follows;
    26 C     CALL UNIVENT( X, NX, THRESH, LTHR, IC )
    27 C
    28 C*********************************************************************
    29
    30          DIMENSION THRESH(*)
    31          INTEGER*2 INDEX(*)
    32
    33 C     NOTE:  The first X-dimension (below) MUST be exactly the same as
    34 C            as in the calling program, even if NX may be different!
    35
    36          DIMENSION  X(8,*)
    37
    38 C     If your data is integer, remove the comment from column 1
    39 C     of the next line of code.
    40 C     INTEGER DATA,DATAT
    41
    42
    43          NSAMPLE = IC * 2 ** NX
    44          NTHRESH = 2 ** NX - 1
    45 C
    46 C     Initialize the index array.
    47 C
    48          IF(LDBUG.GE.1) PRINT *, 'NSAMPLE ',NSAMPLE,' NTHRESH ',NTHRESH
    49
    50          DO 10 I = 1 , NSAMPLE
    51              INDEX( I) = I
    52    10 CONTINUE
    53
    54 C*********************************************************************
    55 C
    56 C     For each of the NX dimensions, I, of the data vector, we
    57 C     determine the 2**(I-1) thresholds which divide the space into
    58 C     approximately equal (based on the sample) probability bins given
    59 C     that we have already divided the space for all dimensions less
    60 C     than i and we consider for each of the 2**(I-1) thresholds
```

```
61 C        only the sample data in one of the previous 2**(I-1-1) bins.
62 C        For example, if I=1, all of the sample is divided into one of two
63 C        bins based on the value of the 1st component of the NSAMPLE
64 C        NX-vectors.  Based on this 2**(1-1)=1 threshold and the
65 C        value of the 2nd component of the data vectors in each of the two
66 C        subsets of the NSAMPLE data points, these subsets are then divided
67 C        with 2**(2-1)=2 thresholds.  The threshold used is the median of
68 C        the sampled component.
69 C
70 C*********************************************************************************
71
72        NS = 1
73        IC2PNXI  = IC * 2**(NX+1)
74        DO 130 I = 1 , NX
75
76            JSTOP = 0
77            IC2PNXI  = IC2PNXI/2
78            JSTART= 1-IC2PNXI
79
80
81            DO 120 L = 1 , NS
82
83                JSTART = JSTART + IC2PNXI
84                JSTOP  = JSTOP  + IC2PNXI
85
86
87        IF(LDBUG.GE.3) THEN
88                WRITE (6,*) 'JSTART ',JSTART,' JSTOP ',JSTOP,' I,L ',I,L
89                WRITE (6,1002) (INDEX(M), M=1,NSAMPLE ),2**(I-1)+L-1
90        END IF
91
92 C*********************************************************************************
93 C        This code sorts the data array IDATA w.r.t. it's Ith
94 C        component (column) for the data values corresponding to
95 C        INDEX(JSTART),INDEX(JSTART+1),....,INDEX(JSTOP).
96 C*********************************************************************************
97
98                DO 110 J = JSTART+1, JSTOP
99
100                    INDEXT = INDEX (J)
101                    DATAT = X(I,INDEX(J))
102
103                    IF(LDBUG.GE.4)
104      1             WRITE (6,*) 'J ',J,' INDEXT ',INDEXT,' DATAT ',DATAT
105
106                    DO 100 K = J-1, JSTART, -1
107
108                        IF(LDBUG.GE.5) THEN
109                        WRITE (6,*) 'K, DATAT, INDEX(K) ',K,DATAT,INDEX(K)
110                        WRITE (6,*) 'X(I,INDEX(K)) ',X(I,INDEX(K))
111                        END IF
112
113                        IF ( DATAT .LT. X(I,INDEX(K)) ) THEN
114
115                            INDEX(K+1) = INDEX(K)
116                            INDEX(K) = INDEXT
117
118                            IF(LDBUG.GE.5) THEN
119                            WRITE (6,*) 'TEST DATA < DATA ABOVE,BUBBLE UP'
120                            WRITE (6,*) 'J ',J,' INDEX(J) ',INDEX(J)
```

```
121                                    WRITE (6,*) 'K ',K,' INDEX(K) ',INDEX(K)
122                                    END IF
123
124                           ELSE
125
126          IF(LDBUG.GE.5)    WRITE (6,*) 'TEST DATA > DATA ABOVE, NEXT J'
127
128                           GOTO 110
129
130                           ENDIF
131
132   100              CONTINUE
133
134   110        CONTINUE
135
136 C*********************************************************************************
137 C         Now use the sorted data to determine the median/threshold.
138 C*********************************************************************************
139
140              MIDINDEX = INT ( ( JSTOP + JSTART ) / 2 )
141
142 C     If your data is integer, remove the comment from column 1
143 C     of the next three lines of code, and comment out the
144 C     non-floated definition of THRESH.
145 C          THRESH(2**(I-1)+L-1)=
146 C     +     ( FLOAT ( X(I, INDEX(MIDINDEX) ) ) +
147 C     +     FLOAT ( X(I, INDEX(MIDINDEX+1) ) ) ) / 2.
148
149          THRESH(2**(I-1)+L-1)=
150     +     ( X( I, INDEX(MIDINDEX) ) +
151     +       X( I, INDEX(MIDINDEX+1) ) ) / 2.
152
153          IF(LDBUG.GE.3) THEN
154          WRITE (6,*) 'THRESH(',2**(I-1)+L-1,')',
155     +              THRESH(2**(I-1)+L-1)
156          WRITE (6,*) ' MIDINDEX ',MIDINDEX
157
158          WRITE (6,1002) (INDEX(M), M=1,NSAMPLE ),I,2**(I-1)+L-1
159          WRITE (6,1001) (THRESH(M), M=1,NTHRESH )
160          END IF
161
162   120     CONTINUE
163          NS = 2*NS
164
165   130 CONTINUE
166
167      IF(LDBUG.GE.2) THEN
168      WRITE (6,1001) (THRESH(M), M=1,NTHRESH )
169      WRITE (6,1004) ((I,J,X(J,INDEX(I)), J=1,NX ),I=1,NSAMPLE )
170      END IF
171
172
173 1001 FORMAT(10(1X,F7.3))
174 1002 FORMAT(20(1X,I3))
175 1003 FORMAT( F9.0, 11X, 4 ( 1X, I3, 1X, F10.4 ) )
176 1004 FORMAT(5(1X,I3,1X,I3,1X,F7.4))
177
178      RETURN
179      END
```

NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0

NUMBER OF WARNINGS IN COMPILATION : 0
NUMBER OF ERRORS   IN COMPILATION : 0

```
 1              SUBROUTINE GIBBS(X,NX,TILEX,Y,NY,TILEY,NTRIALS,
 2        1                     HISTO,NHX,NHY,INSTR,HX,HY,H,G,LDBUG)
 3
 4 C      This subroutine computes three entropies associated with two
 5 C      random vectors  X  and  Y, of dimensions  NX and NY.  H is the
 6 C      entropy of the concatenated vectors after NTRIALS of the ex-
 7 C      periment.  HX and HY are the separate entropies of  X and Y
 8 C      after NTRIALS.  G = HX+HY-H  is the Gibbs relative entropy
 9 C      of the combined system.  All entropies are computed with res-
10 C      pect to the tiling of the event space specified by the TILE
11 C      arrays.
12
13              DIMENSION X(NX),Y(NY),TILEX(*),TILEY(*)
14              INTEGER*2 HISTO(0:NHX,0:NHY)
15              PARAMETER (ALN2=0.6931471)
16
17              IF(INSTR .EQ. 0) GO TO 310
18              NTRIALS = NTRIALS+1
19
20 C      IDENTIFY THE X-EVENT NUMBER
21
22              KX    = 0
23              LEVEL = 1
24
25              DO 10 J=1,NX
26                  IF( X(J) .GT. TILEX(LEVEL+KX) ) KX = LEVEL + KX
27                  LEVEL = 2*LEVEL
28 10           CONTINUE
29
30 C      IDENTIFY THE Y-EVENT NUMBER
31
32              KY    = 0
33              LEVEL = 1
34
35              DO 20 J=1,NY
36                  IF( Y(J) .GT. TILEY(LEVEL+KY) ) KY = LEVEL + KY
37                  LEVEL = 2*LEVEL
38 20           CONTINUE
39
40                  IF(LDBUG.GE.1) THEN
41                  PRINT *,'X = '
42                  PRINT *,(X(I),I=1,NX)
43                  PRINT *,'   X-EVENT I.D. = ',KX
44                  PRINT *
45                  PRINT *,'Y = '
46                  PRINT *,(Y(I),I=1,NY)
47                  PRINT *,'   Y-EVENT I.D. = ',KY
48                  END IF
49
50 C      BUMP THE COUNT FOR THE IDENTIFIED COMPOSITE EVENT
51
52              HISTO(KX,KY) = HISTO(KX,KY) + 1
53
54              IF( INSTR .EQ. 1 ) RETURN
55
56 310          CONTINUE
57
58 C      COMPUTE THE ENTROPIES ASSOCIATED WITH THE ACCUMULATED HISTOGRAM.
59
60              H = 0.
```

```
 61           HX= 0.
 62           HY= 0.
 63           FLN = FLOAT(NTRIALS)
 64
 65           DO 330 I=0,NHX
 66              MARX = 0
 67           DO 320 J=0,NHY
 68           P = FLOAT(HISTO(I,J))/FLN
 69           IF(P.GT.0) H = H - P*LOG(P)
 70              MARX = MARX+HISTO(I,J)
 71 320       CONTINUE
 72           PX = FLOAT(MARX)/FLN
 73              IF(PX.GT.0) HX = HX - PX*LOG(PX)
 74 330       CONTINUE
 75
 76           DO 350 J=0,NHY
 77              MARY = 0
 78           DO 340 I=0,NHX
 79              MARY = MARY+HISTO(I,J)
 80 340       CONTINUE
 81           PY = FLOAT(MARY)/FLN
 82              IF(PY.GT.0) HY = HY - PY*LOG(PY)
 83 350       CONTINUE
 84   .
 85 C         COMPUTE THE NORMALIZED STRUCTURE FUNCTION
 86 C         G = 2.*(HX+HY - H)/((NX+NY)*ALN2)    [ SUPERCEDED ]
 87           HX = HX/ALN2
 88           HY = HY/ALN2
 89           H  = H /ALN2
 90           SUPHXHY = FLOAT(NX+NY)
 91           AMINH   = AMAX1(HX,HY)
 92           G = (HX+HY-H)/(SUPHXHY-AMINH)
 93
 94           IF( INSTR .EQ. 2 ) RETURN
 95
 96 C         RESET THE HISTOGRAM TO ZERO FOR THE NEXT EXPERIMENT.
 97
 98           DO 420 I=0,NHX
 99           DO 410 J=0,NHY
100           HISTO(I,J) = 0
101 410       CONTINUE
102 420       CONTINUE
103           NTRIALS     = 0
104
105           RETURN
106           END
```

NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0

NUMBER OF WARNINGS IN COMPILATION : 0
NUMBER OF ERRORS   IN COMPILATION : 0

```
    1              SUBROUTINE NAMELIST(IOUNIT,N1,N2,N3)
    2 C
    3 C          The following declarations are for local variables.
    4              CHARACTER      LINE*72,NAME*8,NVAL*72
    5 C
    6 C
    7 C          This program is intended to approximate the NAMELIST capability
    8 C          which some FORTRAN compilers have, but which RM-FORTRAN does not
    9 C          have.  The calling program needs to have a common block labeled
   10 C          /NLIST/ and containing six arrays:
   11 C
   12              CHARACTER*8     R4NAMES
   13              REAL*4          R4VALUE
   14 C
   15              CHARACTER*8     I4NAMES
   16              INTEGER*4       I4VALUE
   17 C
   18              CHARACTER*8     L1NAMES
   19              LOGICAL*1       L1VALUE
   20 C
   21              COMMON /NLIST/ R4NAMES(21),I4NAMES(21),L1NAMES(7),
   22         1                   R4VALUE(21),I4VALUE(21),L1VALUE(7)
   23 C
   24              IF(N1.GT.21.OR.N2.GT.21.OR.N3.GT.7) THEN
   25                 PRINT *,'N1 or N2 or N3 is too large for NAMELIST.'
   26                 PRINT *,'Increase dimensions in /NLIST/ common, and'
   27                 print *,'increase limits in first statement of NAMELIST.'
   28                 STOP
   29              END IF
   30 C
   31 1            CONTINUE
   32 C
   33              IF(IOUNIT.EQ.6) THEN
   34              PRINT *,'ENTER VARIABLE NAMES FOLLOWED BY VALUES ACCORDING TO'
   35              PRINT *,'THE SYNTAX, name = value <CR>.  SPACES ARE OPTIONAL.'
   36              PRINT *,'LEGAL NAMES AND CURRENT VALUES ARE:'
   37              PRINT *
   38              WRITE(6,2) (R4NAMES(J),R4VALUE(J),J=1,N1)
   39              WRITE(6,3) (I4NAMES(J),I4VALUE(J),J=1,N2)
   40              WRITE(6,4) (L1NAMES(J),L1VALUE(J),J=1,N3)
   41 2            FORMAT((4(2X,A8,'[',F8.3,']')))
   42 3            FORMAT((4(2X,A8,'[ ',I6,' ]')))
   43 4            FORMAT((6(2X,A8,'[ ',L1,' ]')))
   44              PRINT *
   45              PRINT *,'IF YOU GOOF, JUST RE-ENTER THE LINE CORRECTLY.'
   46              PRINT *,'ANY LINE NOT HAVING THE = SIGN IN IT TERMINATES ENTRY,'
   47              PRINT *,'EXCEPT "?" DISPLAYS VALUES AND "#" STOPS THE PROGRAM.'
   48              PRINT *
   49              END IF
   50 C
   51 10           CONTINUE
   52 C
   53              READ(IOUNIT,12,END=100) LINE
   54 12           FORMAT(A72)
   55              IF(LINE.EQ.'#') STOP ' **** User STOP in NAMELIST'
   56              IF(LINE.EQ.'?') GOTO 1
   57              NEQ = INDEX(LINE,'=')
   58              IF(NEQ.EQ.0) GOTO 100
   59 C             IF(LINE.EQ.'QUIT') THEN
   60 C                  LINE='QUIT=T'
```

```
 61 C                GOTO 13
 62 C             END IF
 63 C             GOTO 100
 64 C          END IF
 65
 66 13     CONTINUE
 67
 68        NAME=LINE(1:NEQ-1)
 69        NVAL=LINE(NEQ+1:72)
 70 C
 71        DO 20 I=1,N1
 72        IF(NAME.EQ.R4NAMES(I)) THEN
 73          IF(INDEX(NVAL,'.').EQ.0) THEN
 74              NPT = INDEX(NVAL,'          ')
 75              NVAL(NPT:NPT) = '.'
 76          END IF
 77          READ(NVAL,15) R4VALUE(I)
 78 15       FORMAT(F15.5)
 79 C        PRINT *, R4NAMES(I),' = ',R4VALUE(I)
 80 C        PRINT *
 81          GO TO 10
 82        END IF
 83 20     CONTINUE
 84 C
 85        DO 30 I=1,N2
 86        IF(NAME.EQ.I4NAMES(I)) THEN
 87          READ(NVAL,25) I4VALUE(I)
 88 25       FORMAT(I15)
 89 C        PRINT *, I4NAMES(I),' = ',I4VALUE(I)
 90 C        PRINT *
 91          GO TO 10
 92        END IF
 93 30     CONTINUE
 94 C
 95        DO 40 I=1,N3
 96        IF(NAME.EQ.L1NAMES(I)) THEN
 97 32       CONTINUE
 98          NDXT = INDEX(NVAL,'T')
 99          NDXF = INDEX(NVAL,'F')
100          IF(NDXT*NDXF .NE. 0 .OR. (NDXT+NDXF).EQ. 0) THEN
101             PRINT *,NAME,'IS A LOGICAL VARIABLE.   ENTER T OR F > '
102             READ(6,12) NVAL
103             GO TO 32
104          END IF
105          IF(NDXT.NE.0) NVAL='.TRUE.'
106          IF(NDXF.NE.0) NVAL='.FALSE.'
107          READ(NVAL,35) L1VALUE(I)
108 35       FORMAT(L15)
109 C        PRINT *, L1NAMES(I),' = ',L1VALUE(I)
110 C        PRINT *
111          GO TO 10
112        END IF
113 40     CONTINUE
114 C
115        PRINT *,'VARIABLE NAME  ',NAME,'  NOT RECOGNIZED.'
116        PRINT *,'INPUT CONTINUES...'
117        GO TO 1
118 C
119 100    CONTINUE
120        PRINT *,'User input complete.'
```

```
121        PRINT *
122        RETURN
123        END
```

NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0


```
124
```

NUMBER OF WARNINGS IN COMPILATION : 0
NUMBER OF ERRORS   IN COMPILATION : 0

```
 1 C*              ***********************************************
 2 C*              *                                             *
 3 C*              *               H A R R Y N E T               *
 4 C*              *                                             *
 5 C*              *   A RECONFIGURABLE  80-NEURON NETWORK MODEL  *
 6 C*              *   WHICH LEARNS BY THE DRIVE-REINFORCEMENT LAW *
 7 C*              ***********************************************
 8 C*
 9 C*
10         SUBROUTINE HARRYNET(T,KSROOT,KIROOT,KOROOT)
11         INCLUDE '\SYSPRO\COMNSH.INC'
12 C******
13 C      \SYSPRO\COMNSH.INC -- Abbreviated labeled common arrays, for use
14 C                              in all subroutines except EVOLVE.
15 C      NEVER CHANGE ANYTHING IN THIS FILE.
16 C      Use an INCLUDE statement to use these common arrays in any
17 C      SYSPRO subroutine.
18 C
19         COMMON   /STATSP/   STATEV(   1)
20         COMMON   /KSNAME/   KSNAME(2,  1)
21         COMMON   /INPSP /   RINPUT(   1)
22         COMMON   /KINAME/   KINAME(2,  1)
23         COMMON   /OUTPSP/   OUTPUT(   1)
24         COMMON   /KONAME/   KONAME(2,  1)
25         COMMON   /OUTINT/   OUTINT(2,  1)
26         COMMON   /TIME  /   TIME
27         CHARACTER*12 KSNAME,KINAME,KONAME,ISYSNM*6
28         COMMON   /SIMVAR/   ENDTIM,MODE,DELTAT,TIMINC,NPRINT,AUDIT,RANDOM,
29        1                    NSYS,NXTSUB,ISYS(7,110),ISUB(0:220),ISYSNM(110),
30        1                    NPLOTS,NSKIP,KURVE(5,51),NPAGE,RSMIN,RSMAX,RSEED
31         LOGICAL             AUDIT,RANDOM
32         COMMON  / DTG  /    ISEC,IMIN,IHR,IDAY,IMO,IYR,
33        1                     JSEC,JMIN,JHR,JDAY,JMO,JYR,
34        1                     KSEC,KMIN,KHR,KDAY,KMO,KYR
35         COMMON   /TITLE /   ITITLE(40,5),IDATE,ITIME
36         CHARACTER           ITITLE*2,IDATE*9,ITIME*8
37 C
38 C****** END OF \SYSPRO\COMNSH.INC
39         INCLUDE '\BPNET\RUMDAT.INC'
40         COMMON /RUMDAT/ GAMMA(4),PARMTHR(5,0:4)
41         COMMON /NETWORK/ NUMINPT,INPUT(50),NUMNEUR,NEURN(100),
42        1                 NEDGE(2,5100),EDGEWT(5100),NFANIN(4,100)
43 C
44 C***
45 C         COMMON      /WORKSP/ W(20,9)
46 C*                            ^^        EXACTLY AS IN MAIN PROGRAM.
47 C
48 C         VARIABLE NAMING SECTION
49 C         ***********************
50 C
51         CHARACTER*12 LSYSNM
52         CHARACTER*12 LSNAME(2,  1), LINAME(2,50), LONAME(2,50)
53 C*                            ^^             ^^             ^^
54 C*                          = KSLEN        = KILEN        = KOLEN
55 C
56         DIMENSION       KVNDX( 3, 0:80)
57 C
58 C      THE NUMBER OF SUBSYSTEMS, N = NSUBS, IS:
59 C*                              v
60         DATA            NSUBS /  80/
```

```
 61 C
 62 C         THE LENGTHS OF THE "SYSMODEL" SYSTEM VECTORS (EXCLUDING
 63 C         SYS1 - SYSN) ARE:
 64 C
 65 C*           STATEV ::    KSLEN = vv   ( MUST = O FOR COMPOSITE SYSTEM)
 66             DATA          KSLEN /  0/
 67 C*           RINPUT ::    KILEN = vv
 68             DATA          KILEN / 50/
 69 C*           OUTPUT ::    KOLEN = vv
 70             DATA          KOLEN / 50/
 71 C
 72 C         THESE VALUES ARE REPORTED TO THE CALLING PROGRAM BY THE
 73 C         "AUDIT" SECTION IF  AUDIT = .TRUE.
 74 C
 75 C         IF  N>1   THEN THIS IS A COMPOSITE SYSTEM AND IT EVOLVES THE
 76 C         STATEV  INDIRECTLY BY FIRST EXECUTING THE CROSSTALK FUNCTIONS
 77 C         (Q1,...,QN) TO ADJUST THE INPUTS TO THE SUBSYSTEMS AND THEN
 78 C         BY CALLING THE SUBSYSTEM MODELS (SYS1,....,SYSN).
 79 C
 80 C         INTERMEDIATE VALUES (NOT REQUIRED TO BE KNOWN UPON ANY ENTRY
 81 C         INTO THIS  SYSMODEL  SUBROUTINE) SHOULD BE EQUIVALENCED TO
 82 C         THE WORKSPACE VECTOR,  W , TO SAVE SPACE.   DO THIS NOW:
 83 C
 84 C*        EQUIVALENCE      ( W( 1,1), TEMP1 )
 85 C*                         (ETC.)
 86 C
 87 C                         STATE   INPUT   OUTPUT
 88             DATA    KVNDX/ 1,1,1,1,    51.        51,
 89      1                          237*0              /
 90 C         NOTE: The remaining components of the KVNDX array will be com-
 91 C               puted in the AUDIT SECTION below, on the assumption that
 92 C               the NEURON subsystems each have 60 statevector components,
 93 C               60 inputvector components, and 6 outputvector components.
 94 C
 95 C         SYSTEM NAME:
 96             DATA LSYSNM/                  'KLOPF'                 /
 97 C*                                        ^^^^^
 98 C
 99 C         THE LENGTH OF THE HISTORY SEGMENT FOR EACH SYNAPSE IS:
100             DATA            LHIST/6/
101
102
103 C         STATEMENT FUNCTION SECTION
104 C         **************************
105 C
106 C         NDS(J) IS THE INDEX OF THE  J-TH  ENTRY OF THE STATE VECTOR
107 C         OF THIS SYSTEM (I.E., RELATIVE TO KSROOT), AND SIMILARLY FOR
108 C         NDI(J) AND NDO(J).
109 C
110             NDS(J) = J + KSROOT
111             NDI(J) = J + KIROOT
112             NDO(J) = J + KOROCT
113 C
114 C         NRS(I) IS THE INDEX OF THE ROOT OF THE STATE VECTOR OF THE
115 C         I-TH SUBSYSTEM OF THIS  SYSMODEL. (ETC. FOR NRI, NRO)
116 C
117             NRS(I) = KSROOT + KVNDX(1,I) - 1
118             NRI(I) = KIROOT + KVNDX(2,I) - 1
119             NRO(I) = KOROOT + KVNDX(3,I) - 1
120 C
```

```
121 C          THE FOLLOWING STATEMENT FUNCTIONS SIMPLIFY REFERENCES TO THE
122 C          SYSTEM VECTOR ELEMENTS.   THEY MAY BE USED ONLY ON THE RIGHT
123 C          SIDE OF AN ASSIGNMENT STATEMENT.   ON THE LEFT SIDE OF AN
124 C          ASSIGNMENT STATEMENT, THE FULL REFERENCE MUST BE USED.
125 C
126            ST(I)  = STATEV(NDS(I))
127            RI(I)  = RINPUT(NDI(I))
128            OU(I)  = OUTPUT(NDO(I))
129            STSUB(J,I) = STATEV(NRS(J)+I)
130            OUSUB(J,I) = OUTPUT(NRO(J)+I)
131 C
132 C                              ******************
133 C
134 C          AUDIT SECTION
135 C          *************
136 C
137 1000       CONTINUE
138            IF(.NOT. AUDIT) GO TO 2000
139 C
140            IF(NUMNEUR.GT.NSUBS .OR. NUMNEUR.LT.1) GOTO 5100
141 C
142            NSUBS = NUMNEUR
143            DO 1010 I=2,NSUBS
144               KVNDX(1,I) = KVNDX(1,1) + (I-1)* 60
145               KVNDX(2,I) = KVNDX(2,1) + (I-1)* 63
146               KVNDX(3,I) = KVNDX(3,1) + (I-1)* 7
147 1010       CONTINUE
148 C
149 C          INITIALIZE SYSTEM VECTOR LABELS
150            DO 1020 J=1,50
151            WRITE(LINAME(1,J),1091) J
152 1091       FORMAT('NET ',I2,' INPUT')
153            WRITE(LINAME(2,J),1093)
154 1093       FORMAT('FIRING RATE ')
155            WRITE(LONAME(1,J),1095) J
156 1095       FORMAT('NET ',I2,' OUTPT')
157            WRITE(LONAME(2,J),1096)
158 1096       FORMAT('SIGNAL       ')
159 1020       CONTINUE
160 C
161            CALL SYSAUD(NSUBS,KSROOT,KIROOT,KOROOT,KSLEN,KILEN,KOLEN,
162         1            LSYSNM,LSNAME,LINAME,LONAME)
163 C
164 C          SKIP THE SUBSYSTEM CROSSTALK SECTION DURING AUDIT.
165            GO TO 3000
166 C
167 C                              ******************
168 C
169 C          SUBSYSTEM CROSSTALK SECTION
170 C          ***************************
171 2000       CONTINUE
172 C
173 C          DISTRIBUTE EXTERNAL INPUTS TO THEIR DESIGNATED SYNAPSES.
174 C
175            DO 2100 J = 1,NUMINPT
176                   Y = RI(J)
177                   IEDGE = INPUT(J)
178                   N = NEDGE(1,IEDGE)
179                   IF(N.EQ.0 .OR. IEDGE.EQ.0) GO TO 2100
180                   DO 2050 K = IEDGE+1,IEDGE+N
```

```
181                         NN       = NEDGE(1,K)
182 C                                = DESTINATION NEURON #
183                         KSYN     = NEDGE(2,K)
184 C                                = DESTINATION SYNAPSE # ON DEST'N NEURON
185                         IF(KSYN .EQ. 0) THEN
186                         OUTPUT(NRO(NN)+2) = Y
187                         ELSE
188 C                   SHIFT PRIOR INPUTS TO RIGHT
189                        JSYN = NRI(NN) + (KSYN-1)*(LHIST+1)
190                        DO 2040 JL = LHIST,1,-1
191                         JPL = JSYN + JL + 1
192                         RINPUT(JPL) = RI(JPL-1)
193 2040                    CONTINUE
194                        RINPUT(JSYN+1) = Y*EDGEWT(K)
195                         END IF
196 2050    CONTINUE
197 2100    CONTINUE
198 C
199 C    WARNING:  The /NETWORK/ common block violates the system simulation
200 C              rules.  This network cannot be assembled into a larger
201 C              system.  Fold it into /RINPUT/ before trying to include
202 C              BPNET into any larger SYSPRO system.
203 C
204 C                              *******************
205 C
206 C       STATE EVOLUTION SECTION
207 C       ***********************
208 3000    CONTINUE
209 C
210 C
211         DO 3010 J = 1,NUMNEUR
212             CALL KLOPFON(T,NRS(J),NRI(J),NRO(J))
213 3010    CONTINUE
214 C
215         IF( AUDIT )  RETURN
216 C*      GO TO 4000
217 C
218 C                              *************************
219 C
220 C       READOUT SECTION
221 C       ***************
222 4000    CONTINUE
223 C
224 C    NOTE:   EACH SUBSYSTEM HAS ITS OWN READOUTS.   THE ONLY READOUTS
225 C            THAT SHOULD BE INCLUDED HERE ARE THOSE THAT USE  STATEV
226 C            COMPONENTS WHICH ARE NOT ALL MEMBERS OF A SINGLE SUB-
227 C            SYSTEM STATE VECTOR.
228 C
229 C    DISTRIBUTE EACH NEURON'S OUTPUTS TO THEIR DESIGNATED SYNAPSES
230 C
231         DO 4200 J = 1,NUMNEUR
232             IEDGE = NEURN(J)
233             N = NEDGE(1,IEDGE)
234             IF(IEDGE.EQ.0 .OR. N.EQ.0) GO TO 4200
235             DO 4150 K = IEDGE+1,IEDGE+N
236                 IF(NEDGE(1,K).NE.0) THEN
237                     DO 4130 L=1,LHIST+1
238                      KL = NRI(NEDGE(1,K))+NEDGE(2,K)+L-1
239                      RINPUT(KL) = OUSUB(J,L)*EDGEWT(K)
240 4130                CONTINUE
```

```
 241                ELSE
 242                   OUTPUT(NDO(NEDGE(2,K))) = OUSUB(J,1)
 243                END IF
 244 4150       CONTINUE
 245 4200    CONTINUE
 246 C
 247 C
 248 4999    RETURN
 249 C       ******
 250 C
 251 C       ERROR RECOVERY SECTION
 252 C       **********************
 253 5000    CONTINUE
 254 C
 255 5100    PRINT 5900, NUMNEUR,NSUBS
 256         STOP
 257 5900    FORMAT(' SUBROUTINE BPNET - ERROR:  YOU HAVE ',I4,' NEURONS.'/
 258      1 '                         ARRAY DIMENSIONS ONLY ALLOW ',I4,'.',/)
 259 C
 260         END
```

```
NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0


NUMBER OF WARNINGS IN COMPILATION : 0
NUMBER OF ERRORS   IN COMPILATION : 0
```

```
 1 C*          ********************************************************
 2 C*          *                                                      *
 3 C*          *                  K L O P F O N                       *
 4 C*          *                                                      *
 5 C*          *   BASIC PROCESSING ELEMENT MODEL FOR DRIVE-          *
 6 C*          *   REINFORCEMENT NEURON MODEL FOR BIOMASSCOMP         *
 7 C*          ********************************************************
 8 C*                           JANUARY, 1988
 9 C*
10           SUBROUTINE KLOPFON(T,KSROOT,KIROOT,KOROOT)
11 C***
12           INCLUDE '\SYSPRO\COMNSH.INC'
13 C******
14 C      \SYSPRO\COMNSH.INC -- Abbreviated labeled common arrays, for use
15 C                              in all subroutines except EVOLVE.
16 C      NEVER CHANGE ANYTHING IN THIS FILE.
17 C      Use an INCLUDE statement to use these common arrays in any
18 C      SYSPRO subroutine.
19 C
20           COMMON   /STATSP/  STATEV(   1)
21           COMMON   /KSNAME/  KSNAME(2, 1)
22           COMMON   /INPSP /  RINPUT(   1)
23           COMMON   /KINAME/  KINAME(2, 1)
24           COMMON   /OUTPSP/  OUTPUT(   1)
25           COMMON   /KONAME/  KONAME(2, 1)
26           COMMON   /OUTINT/  OUTINT(2, 1)
27           COMMON   /TIME /   TIME
28           CHARACTER*12 KSNAME,KINAME,KONAME,ISYSNM*6
29           COMMON   /SIMVAR/  ENDTIM,MODE,DELTAT,TIMINC,NPRINT,AUDIT,RANDOM,
30          1                   NSYS,NXTSUB,ISYS(7,110),ISUB(0:220),ISYSNM(110),
31          1                   NPLOTS,NSKIP,KURVE(5,51),NPAGE,RSMIN,RSMAX,RSEED
32           LOGICAL            AUDIT,RANDOM
33           COMMON   / DTG /   ISEC,IMIN,IHR,IDAY,IMO,IYR,
34          1                   JSEC,JMIN,JHR,JDAY,JMO,JYR,
35          1                   KSEC,KMIN,KHR,KDAY,KMO,KYR
36           COMMON   /TITLE /  ITITLE(40,5),IDATE,ITIME
37           CHARACTER          ITITLE*2,IDATE*9,ITIME*8
38 C
39 C****** END OF \SYSPRO\COMNSH.INC
40           INCLUDE '\BPNET\RUMDAT.INC'
41           COMMON /RUMDAT/ GAMMA(4),PARMTHR(5,0:4)
42           COMMON /NETWORK/ NUMINPT,INPUT(50),NUMNEUR,NEURN(100),
43          1                 NEDGE(2,5100),EDGEWT(5100),NFANIN(4,100)
44 C
45 C***
46 C*        COMMON      /WORKSP/  W(20,9)
47 C*                             ^^        EXACTLY AS IN MAIN PROGRAM.
48           DIMENSION  C(5)
49           CHARACTER*1 LETTER(0:5)
50 C
51 C
52 C      VARIABLE NAMING SECTION
53 C      ***********************
54 C
55           CHARACTER*12 LSYSNM
56           CHARACTER*12 LSNAME(2,60), LINAME(2,63), LONAME(2, 7)
57 C*                             ^^            ^^            ^^
58 C*                           = KSLEN       = KILEN       = KOLEN
59 C
60           DIMENSION          KVNDX( 3,  1)
```

```
 61 C*                                ^^^ THIS NUMBER MUST = MAX(1,NSUBS)
 62 C
 63 C*       EXTERNAL   DIFFEQ
 64 C*                  ... AND ANY OTHER EXTERNAL DECLARATIONS.
 65 C
 66 C        THE NUMBER OF SUBSYSTEMS, N = NSUBS, IS:
 67 C*                                     v
 68          DATA                NSUBS /   0/
 69 C
 70 C        THE LENGTHS OF THE "NEURON" SYSTEM VECTORS (EXCLUDING
 71 C        SYS1 - SYSN) ARE:
 72 C
 73 C*           STATEV ::   KSLEN = vv    (TEN SYNAPSES, 6 HISTORICAL VALS)
 74          DATA            KSLEN / 60/
 75 C*           RINPUT ::   KILEN = vv
 76          DATA            KILEN / 63/
 77 C*           OUTPUT ::   KOLEN = vv
 78          DATA            KOLEN /  7/
 79 C
 80 C        THESE VALUES ARE REPORTED TO THE CALLING PROGRAM BY THE
 81 C        "AUDIT" SECTION IF  AUDIT = .TRUE.
 82 C
 83          DATA            LETTER /'a','b','c','d','e','f'/
 84          DATA            LHIST  /6/
 85 C
 86 C             ***********   ***********
 87 C
 88 C        STATEMENT FUNCTION SECTION
 89 C        ***************************
 90 C
 91 C        NDS(J) IS THE INDEX OF THE  J-TH  ENTRY OF THE STATE VECTOR
 92 C        OF THIS SYSTEM (I.E., RELATIVE TO KSROOT), AND SIMILARLY FOR
 93 C        NDI(J) AND NDO(J).
 94 C
 95          NDS(J) = J + KSROOT
 96          NDI(J) = J + KIROOT
 97          NDO(J) = J + KOROOT
 98 C
 99 C
100          ST(I)  = STATEV(NDS(I))
101          RI(I)  = RINPUT(NDI(I))
102          OU(I)  = OUTPUT(NDO(I))
103
104
105 C        AUDIT SECTION   (Calculations needed only on first entry can
106 C        *************    also be inserted here.)
107 C
108 1000     CONTINUE
109          IF(.NOT. AUDIT) GO TO 2000
110 C
111 C        INITIALIZE THE SYSTEM VECTOR LABELS
112          NN = 1+KSROOT/KSLEN
113          WRITE(LSYSNM,1090) NN
114 1090     FORMAT('PE #',I2)
115 C
116          DO 1010 I=1,9
117             IS = 1 + (I-1)*LHIST
118             II = 1 + (I-1)*(LHIST+1)
119          DO 1009 J=0,LHIST
120
```

```
121          IF(J.LT.LHIST) WRITE(LSNAME(1,IS+J),1091) I,LETTER(J),NN
122 1091     FORMAT('WT ',I1,A1,'  PE ',I2)
123          WRITE(LINAME(1,II+J),1092) I,LETTER(J),NN
124 1092     FORMAT('INP ',I1,A1,' PE ',I2)
125          IF(J.LT.LHIST) WRITE(LSNAME(2,IS+J),1093)
126          WRITE(LINAME(2,II+J),1094)
127 1093     FORMAT('D-R SYNAPSE ')
128 1094     FORMAT('FREQUENCY   ')
129
130          IF(I.EQ.1) THEN
131              WRITE(LONAME(1,I+J),1095) NN,J+1
132          END IF
133 1095     FORMAT('PE ',I2,' OUT(',I1,')')
134
135 1009     CONTINUE
136 1010     CONTINUE
137 C
138 C
139          CALL SYSAUD(NSUBS,KSROOT,KIROOT,KOROOT,KSLEN,KILEN,KOLEN,
140        1             LSYSNM,LSNAME,LINAME,LONAME)
141 C
142 C    INITIALIZE SOME QUANTITIES THAT DON'T CHANGE FROM ONE NEURON
143 C    TO THE NEXT:
144
145          THETA = PARMTHR(1,0)
146 C              = FIRING THRESHOLD FOR ALL NEURONS
147          DO 1200 I = 1,5
148           C(I) = PARMTHR(2,I-1)
149 1200     CONTINUE
150 C              = LEARNING RATE CONSTANTS
151          WMIN  = PARMTHR(3,0)
152 C              = LOWER BOUND FOR ABS(SYNAPTIC WEIGHTS)
153          OUMAX = PARMTHR(4,0)
154 C              = UPPER LIMIT FOR OUTPUT LEVEL
155          GAIN  = PARMTHR(5,0)
156 C              = LEARNING RATE GAIN FACTOR
157
158
159 C    SKIP THE SUBSYSTEM CROSSTALK SECTION IN AUDIT CYCLE
160          GO TO 3000
161 C
162 C                        *****************
163 C
164 2000     CONTINUE
165 C    SUBSYSTEM CROSSTALK SECTION
166 C    ***************************
167 C    < NONE FOR PRIMITIVE SYSTEM >
168 C
169 C
170 3000     CONTINUE
171 C    STATE EVOLUTION SECTION
172 C    ***********************
173 C
174 C    FIRST TIME THROUGH, UNDO SOME OF THE (POSSIBLY RANDOM)
175 C    INITIALIZATION OF THE STATE VECTOR WHICH OCCURS AFTER
176 C    THE AUDIT SEQUENCE, ERGO, CAN'T BE DONE ABOVE.
177
178 C    IF(TIME.LT.TIMINC) THEN
179 C    END IF
180
```

```
181              NN = 1 + KSROOT/KSLEN
182 C                 THIS IS THE NUMBER OF THE CURRENT NEURON.
183 C
184              Y = 0.
185              DO 3100 J = 1,49,LHIST
186              Y = Y + ST(J)*RI(J)
187 3100  CONTINUE
188
189              Y = MIN(OUMAX, MAX(0., Y-THETA))
190              DY= OU(1)-Y
191
192
193 C         LEARNING LAW
194 C         ***********+
195 C
196 C         SYNAPTIC LEARNING
197 C             FOR EACH SYNAPSE (I) ...
198              DO 3300 I= 1,9
199              KSY = LHIST*(I-1)+1
200              KIN = (LHIST+1)*(I-1)+1
201
202 C             INTEGRATE OVER THE LAG (J) TO OBTAIN DELTA W.
203              DW= 0.
204              DO 3200 J = 1,LHIST-1
205 C             (NOTE: Should go to LHIST, but that would require 7 input
206 c                    lags, i.e., one more than there are synapse lags.)
207              DXIJ= MAX(0., RI(KIN+J)-RI(KIN+J+1) )
208 C                 (this implements Klopf's refinement on p. 13)
209              DW = DW + C(J)*ABS(ST(KSY+J))*DXIJ
210 3200  CONTINUE
211              DW = DY*DW*GAIN
212
213 C             SHIFT ALL WEIGHTS TOWARD THE PAST (RIGHT SHIFT)
214              DO 3250 J = LHIST-1,1,-1
215                STATEV(NDS(KSY+J)) = ST(KSY+J-1)
216 3250  CONTINUE
217
218 C             UPDATE THE CURRENT VALUE OF THIS SYNAPSE.
219              WT = ST(KSY) + DW
220              IF(ABS(WT).LT.WMIN) THEN
221                STATEV(NDS(KSY)) = SIGN(WMIN,WT)
222              ELSE
223                STATEV(NDS(KSY)) = WT
224              END IF
225
226 3300  CONTINUE
227
228 C
229 4000     CONTINUE
230 C         OUTPUT SECTION
231 C         *************
232 C
233              DO 4100 J=LHIST+1,2,-1
234 4100     OUTPUT(NDO(J)) = OU(J-1)
235              OUTPUT(NDO(1))     = Y
236
237          RETURN
238 C         ******
239 C
240 C         ERROR RECOVERY SECTION
```

```
 241 C        ***********************
 242 5000     CONTINUE
 243 C
 244 C*        INSERT ERROR RECOVERY CODE AND MESSAGES HERE.
 245 C*        WRITE TO UNIT 5 (TERMINAL) OR UNIT 6 (LOGGING FILE).
 246          GO TO 3000
 247 C
 248          END
```

```
NUMBER OF WARNINGS IN PROGRAM UNIT: O
NUMBER OF ERRORS   IN PROGRAM UNIT: O


NUMBER OF WARNINGS IN COMPILATION : O
NUMBER OF ERRORS   IN COMPILATION : O
```

```
 1            SUBROUTINE HKDAT(T)
 2 C
 3 C        This subroutine provides the initial simulation parameters
 4 C        not found in the initialization TRACE file and provides for
 5 C        the computation or the reading of the time-varying sensory
 6 C        inputs to the neural system.
 7 C
 8            INCLUDE '\SYSPRO\COMNSH.INC'
 9 C******
10 C        \SYSPRO\COMNSH.INC -- Abbreviated labeled common arrays, for use
11 C                              in all subroutines except EVOLVE.
12 C        NEVER CHANGE ANYTHING IN THIS FILE.
13 C        Use an INCLUDE statement to use these common arrays in any
14 C        SYSPRO subroutine.
15 C
16            COMMON   /STATSP/   STATEV(   1)
17            COMMON   /KSNAME/   KSNAME(2,  1)
18            COMMON   /INPSP /   RINPUT(   1)
19            COMMON   /KINAME/   KINAME(2,  1)
20            COMMON   /OUTPSP/   OUTPUT(   1)
21            COMMON   /KONAME/   KONAME(2,  1)
22            COMMON   /OUTINT/   OUTINT(2,  1)
23            COMMON   /TIME  /   TIME
24            CHARACTER*12 KSNAME, KINAME, KONAME, ISYSNM*6
25            COMMON   /SIMVAR/   ENDTIM,MODE,DELTAT,TIMINC,NPRINT,AUDIT,RANDOM,
26       1                       NSYS,NXTSUB,ISYS(7,110),ISUB(0:220),ISYSNM(110),
27       1                       NPLOTS,NSKIP,KURVE(5,51),NPAGE,RSMIN,RSMAX,RSEED
28            LOGICAL             AUDIT,RANDOM
29            COMMON   / DTG  /   ISEC, IMIN, IHR, IDAY, IMO, IYR,
30       1                        JSEC,JMIN,JHR,JDAY,JMO,JYR,
31       1                        KSEC,KMIN,KHR,KDAY,KMO,KYR
32            COMMON   /TITLE /   ITITLE(40,5),IDATE,ITIME
33            CHARACTER           ITITLE*2,IDATE*9,ITIME*8
34 C
35 C****** END OF \SYSPRO\COMNSH.INC
36            INCLUDE '\BPNET\RUMDAT.INC'
37            COMMON /RUMDAT/ GAMMA(4),PARMTHR(5,0:4)
38            COMMON /NETWORK/ NUMINPT,INPUT(50),NUMNEUR,NEURN(100),
39       1                     NEDGE(2,5100),EDGEWT(5100),NFANIN(4,100)
40 C
41 C
42            IF(T.GT.TIME) GOTO 1000
43 C
44 D          PRINT 999, 'SUBROUTINE HKDAT:   READING FILE FORT2'
45 D 999      FORMAT(20X,A40)
46 C
47 C          READ THE SYSTEM SIGMOID PARAMETERS:
48            READ(2,904)
49            READ(2,903) ((PARMTHR(J,I),I=0,4),J=1,5)
50 D          PRINT 903,   ((PARMTHR(J,I),I=0,4),J=1,5)
51 C
52 C          READ THE NETWORK GRAPH STRUCTURE:
53            READ(2,908)
54                  MAXINPUTS=50
55                  MAXNEURNS=80
56 C          READ THE EXTERNAL INPUT DISTRIBUTION...
57 C          *************************************
58            NUMINPT = 0
59            IEDGE = 0
60            M1X = 0
```

```
 61 500        CONTINUE
 62            READ(2,907) M1,M2,M3,R4
 63            IF(M1.EQ.0 .AND. M2.EQ.0 .AND. M3.EQ.0) GO TO 600
 64            IF(R4.EQ.0.) R4=1.
 65            IEDGE = IEDGE + 1
 66            IF(M1.EQ.M1X .OR. M1.EQ.0) THEN
 67                    NEDGE(1,INPUT(M1X)) = NEDGE(1,INPUT(M1X)) + 1
 68                    NEDGE(1,IEDGE) = M2
 69                    NEDGE(2,IEDGE) = M3
 70                    EDGEWT(IEDGE)  = R4
 71            ELSE IF(M1.GT.M1X) THEN
 72                    NUMINPT = MAX(NUMINPT,M1)
 73                    INPUT(M1) = IEDGE
 74                    NEDGE(1,IEDGE) = 1
 75                    IEDGE = IEDGE + 1
 76                    NEDGE(1,IEDGE) = M2
 77                    NEDGE(2,IEDGE) = M3
 78                    EDGEWT(IEDGE)  = R4
 79                    M1X   = M1
 80               ELSE
 81                  PRINT 909
 82                  STOP
 83            END IF
 84            GO TO 500
 85 C
 86 600        CONTINUE
 87 C          READ THE INTERNAL CONNECTION GRAPH STRUCTURE
 88 C          ***************************************************
 89            READ(2,904)
 90            M1X = 0
 91 610        READ(2,907) M1,M2,M3,R4
 92            IF(M1.EQ.0 .AND. M2.EQ.0 .AND. M3.EQ.0) GO TO 700
 93            IF(R4.EQ.0.) R4=1.
 94            IEDGE = IEDGE + 1
 95            IF(M1.EQ.M1X .OR. M1.EQ.0) THEN
 96                    NEDGE(1,NEURN(M1X)) = NEDGE(1,NEURN(M1X)) + 1
 97                    NEDGE(1,IEDGE) = M2
 98                    NEDGE(2,IEDGE) = M3
 99                    EDGEWT(IEDGE)  = R4
100            ELSE IF(M1.GT.M1X) THEN
101                    NEURN(M1) = IEDGE
102                    NEDGE(1,IEDGE) = 1
103                    IEDGE = IEDGE + 1
104                    NEDGE(1,IEDGE) = M2
105                    NEDGE(2,IEDGE) = M3
106                    EDGEWT(IEDGE)  = R4
107                    M1X   = M1
108               ELSE
109                  PRINT 910
110                  STOP
111            END IF
112            GO TO 610
113 C
114 700        CONTINUE
115 C          DETERMINE THE NUMBER OF NEURONS IN THE NETWORK
116            NUMNEUR = M1X
117            DO 710 I=1,M1X
118            DO 705 J=NEURN(I)+1, NEURN(I)+NEDGE(1,NEURN(I))
119 705        NUMNEUR = MAX(NUMNEUR,NEDGE(1,J))
120 710        CONTINUE
```

```
121 C
122 D         PRINT *, 'NUMBER OF INPUTS = ',NUMINPT
123 D         PRINT *, 'NUMBER OF NEURONS= ',NUMNEUR
124 C
125 C         CHECK FOR ERRORS IN THE NETWORK ARCHITECTURE
126           IF(NUMINPT.GT.MAXINPUTS) PRINT 911,NUMINPT,MAXINPUTS
127           IF(NUMNEUR.GT.MAXNEURNS) PRINT 912,NUMNEUR,MAXNEURNS
128           IF(NUMNEUR.GT.MAXNEURNS.OR.NUMINPT.GT.MAXINPUTS) STOP
129           IF(NUMNEUR.LT.M1) THEN
130           PRINT *,' INPUT ERROR.  HKDAT COUNTED TOO FEW NEURONS'
131           STOP
132           END IF
133 C
134           READ(2,904)
135           READ(2,902) (GAMMA(I),I=1,4)
136 D         PRINT 902,  (GAMMA(I),I=1,4)
137           IU = 2
138 800       CONTINUE
139 C         SKIP SIX LINES OF INPUT DATA UNIT (NEXT READ WILL BE ON LINE 8)
140           READ(IU,906)
141 805       CONTINUE
142           READ(IU,901,END=1100) TT,N,G,M1,R1,M2,R2,M3,R3,M4,R4
143           IF(TT.LT.O  .AND.  IU.EQ.2) THEN
144           IU = M1
145           GO TO 805
146           END IF
147 C
148           RETURN
149 C
150 1000      CONTINUE
151 D         PRINT 913, TT
152           IF(T.LT.TT) RETURN
153                   IF(N.GT.0) GAMMA(N)=G
154                   IF(M1.GT.0) RINPUT(M1)=R1
155                   IF(M2.GT.0) RINPUT(M2)=R2
156                   IF(M3.GT.0) RINPUT(M3)=R3
157                   IF(M4.GT.0) RINPUT(M4)=R4
158           READ(IU,901,END=1100) TT,N,G,M1,R1,M2,R2,M3,R3,M4,R4
159           GO TO 1000
160 C
161 1100      CONTINUE
162           TT = 1.0E+38
163           GOTO 1000
164 C
165 901       FORMAT(F9.3,1X,I1,1X,F8.3,1X,4(I3,1X,F10.3,1X))
166 902       FORMAT(30X,4F10.8)
167 903       FORMAT(20X,5F10.8)
168 904       FORMAT(////)
169 905       FORMAT(20X,5I10)
170 906       FORMAT(//////)
171 907       FORMAT(I7,I8,I9,F10.4)
172 908       FORMAT(/////////)
173 909       FORMAT(1X,'SUBROUTINE HKDAT -- ERROR READING FILE FORT2'/
174      1            ' EXTERNAL INPUTS MUST BE LISTED IN ASCENDING ORDER'/
175      2            ' EDIT FORT2 AND RE-RUN THE PROGRAM'//)
176 910       FORMAT(1X,'SUBROUTINE HKDAT -- ERROR READING FILE FORT2'/
177      1            ' OUTPUT NEURONS MUST BE LISTED IN ASCENDING ORDER'/
178      2            ' EDIT FORT2 AND RE-RUN THE PROGRAM'//)
179 911       FORMAT(' SUBROUTINE HKDAT - ERROR:  NUMBER OF INPUTS = ',I3,/
180      1            '                          MAXIMUM # INPUTS = ',I3,//)
```

```
181 912      FORMAT(' SUBROUTINE HKDAT  ERROR:  NUMBER OF NEURONS = ',I3,/
182     1                              MAXIMUM # NEURONS = ',I3,//)
183 913      FORMAT(' HKDAT:  READING INPUT DATA FOR TIME = ',F9.3)
184          END
```


NUMBER OF WARNINGS IN PROGRAM UNIT: 0
NUMBER OF ERRORS   IN PROGRAM UNIT: 0

NUMBER OF WARNINGS IN COMPILATION : 0
NUMBER OF ERRORS   IN COMPILATION : 0

APPENDIX   C

DMORPH EXPERIMENTS AND GRAPHS

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   1
DATE: 1/11/88

INPUT CONDITIONS: Two random variables with two components
each. No correlations between the variables.

VARIABLES:

SEED: .247E+13   LEXP: 6000
NX: 2   NY: 2   IC: 8
A: 0.0   B: 1.0   IFUN: 0

X ENTROPY: 1.9574733   Y ENTROPY: 1.9515425
DMORPH: 0.0011546   WHOLE ENTROPY: 3.9066575

COMMENTS:
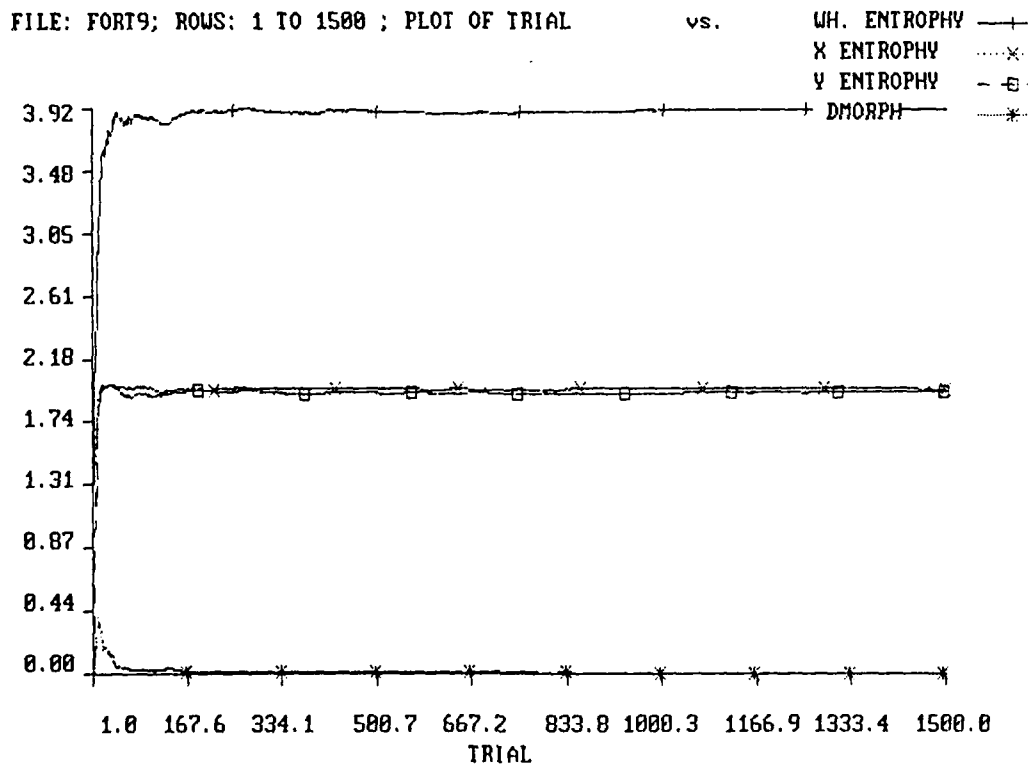
FILE: FORT9; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ——+—
                                                            X ENTROPHY ·····×·
                                                            Y ENTROPHY — ⊟·
                                                            DMORPH———   ·······*·

Page C.1

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   2
DATE: 1/13/88

INPUT CONDITIONS: Two random variables with four components
each. No correlations between the variables.

VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 0

X ENTROPY: 3.86821   Y ENTROPY: 3.96083
DMORPH: 0.01346   WHOLE ENTROPY: 7.77527

COMMENTS:

FILE: EXP2.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ──┼──
                                                          X ENTROPHY ···×··
                                                          Y ENTROPHY ─ ⊟·
                                                          DMORPH   ···*··
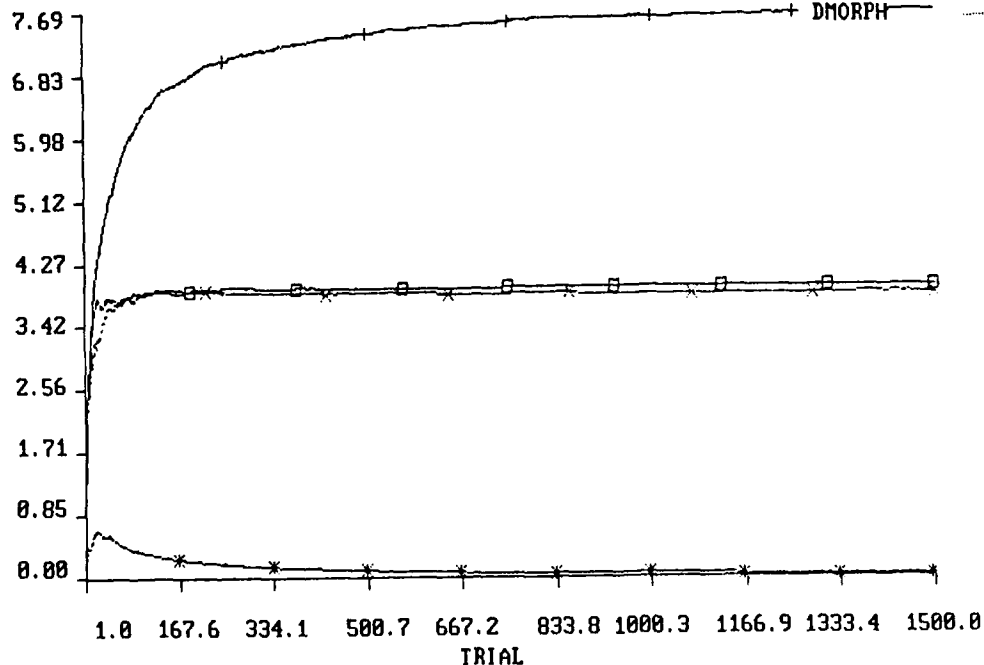
APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT:	BIOMASSCOMP
EXPERIMENT:	DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:	3
DATE:	1/13/88

INPUT CONDITIONS: Two random variables with six components each. No correlations between the variables.

VARIABLES:

SEED:	.247E+13	LEXP:	3000		
NX:	6	NY:	6	IC:	8
A:	0.0	B:	1.0	IFUN:	0

X ENTROPY:	5.87292	Y ENTROPY:	5.89273
DMORPH:	0.30636	WHOLE ENTROPY:	9.89462

COMMENTS:
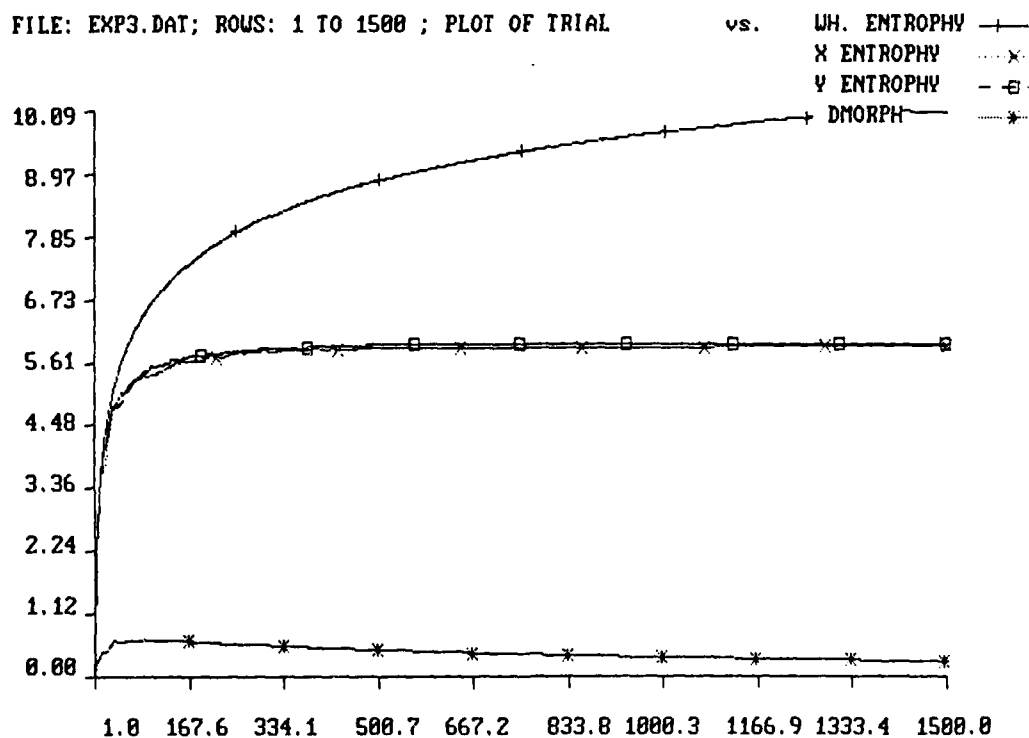
FILE: EXP3.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY —+—

X ENTROPHY ····×·

Y ENTROPHY – ⊟ ·

DMORPH—— ····*··

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   4
DATE: 1/13/88

INPUT CONDITIONS: Two random vectors with eight components
each. No correlations between the variables.

VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 8   NY: 8   IC: 8
A: 0.0   B: 1.0   IFUN: 0

X ENTROPY: 7.82463   Y ENTROPY: 7.82070
DMORPH: 0.53758   WHOLE ENTROPY: 11.26581

COMMENTS:

FILE: EXP4.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ——+—
                                                             X ENTROPHY ·····×·
                                                             Y ENTROPHY - -□-·
                                                             DMORPH—— ·····*··
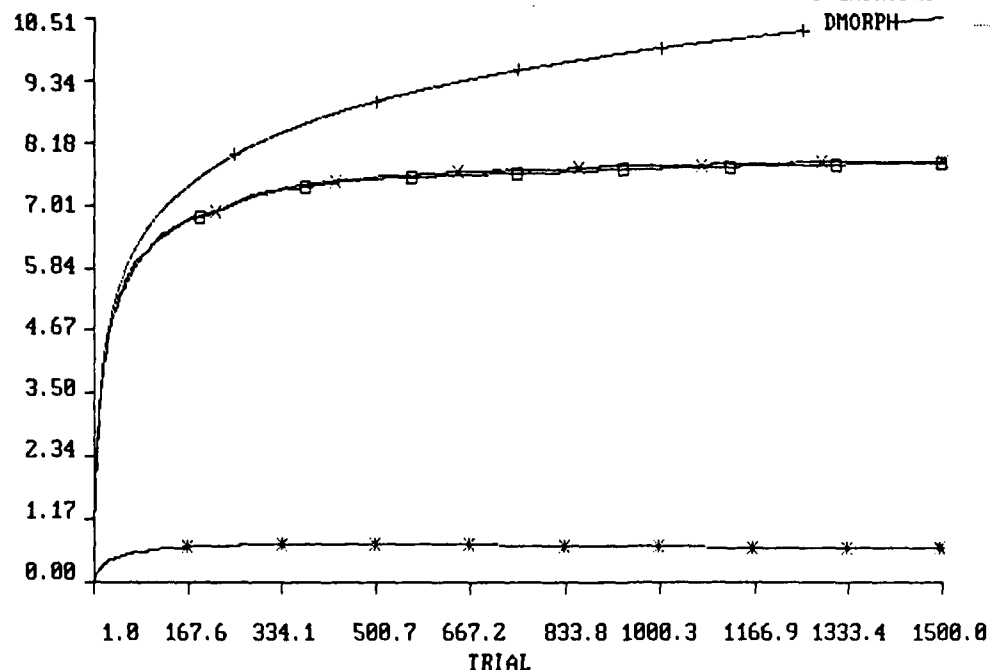
Page C.4

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization
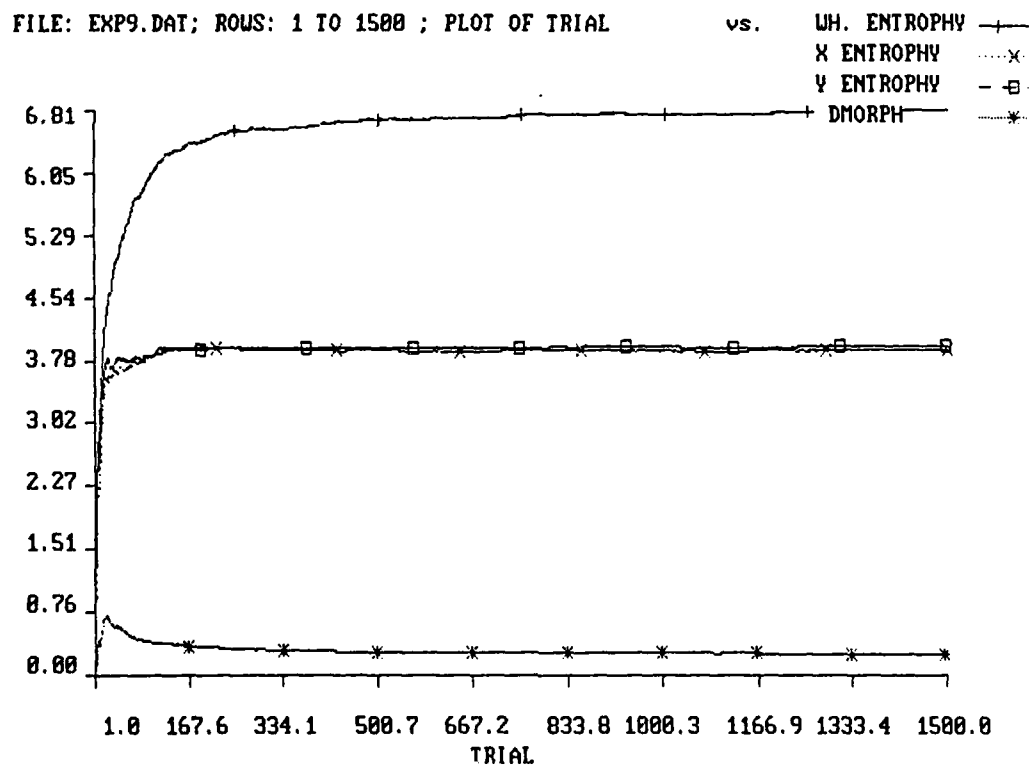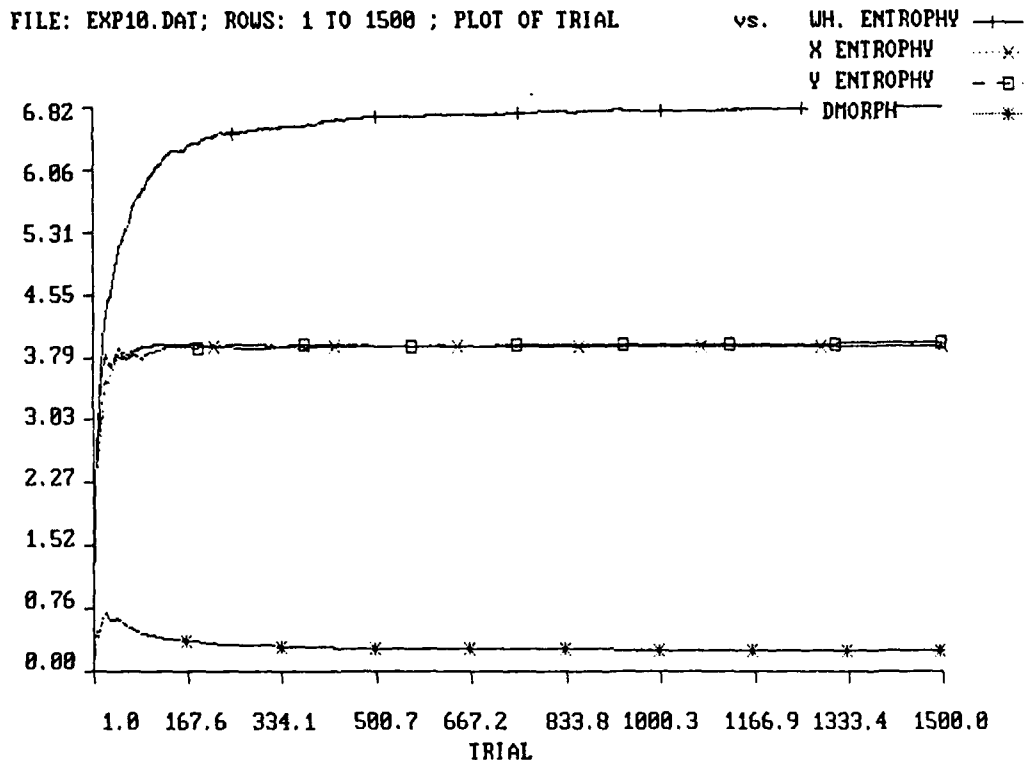
RESEARCHER: David G. Boney

EXPERIMENT #:   5
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. Intravariable correlation, x(1) = x(2).

VARIABLES:

SEED:	.247E+13	LEXP:	3000		
NX:	4	NY:	4	IC:	8
A:	0.0	B:	1.0	IFUN:	1

X ENTROPY:	3.8682065	Y ENTROPY:	3.9612269
DMORPH:	0.013411	WHOLE ENTROPY:	7.7752690

COMMENTS:

FILE: EXP5.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY —+—
                                                           X ENTROPHY ·····×·
                                                           Y ENTROPHY – ⊟·
                                                           DMORPH—— ·····*·
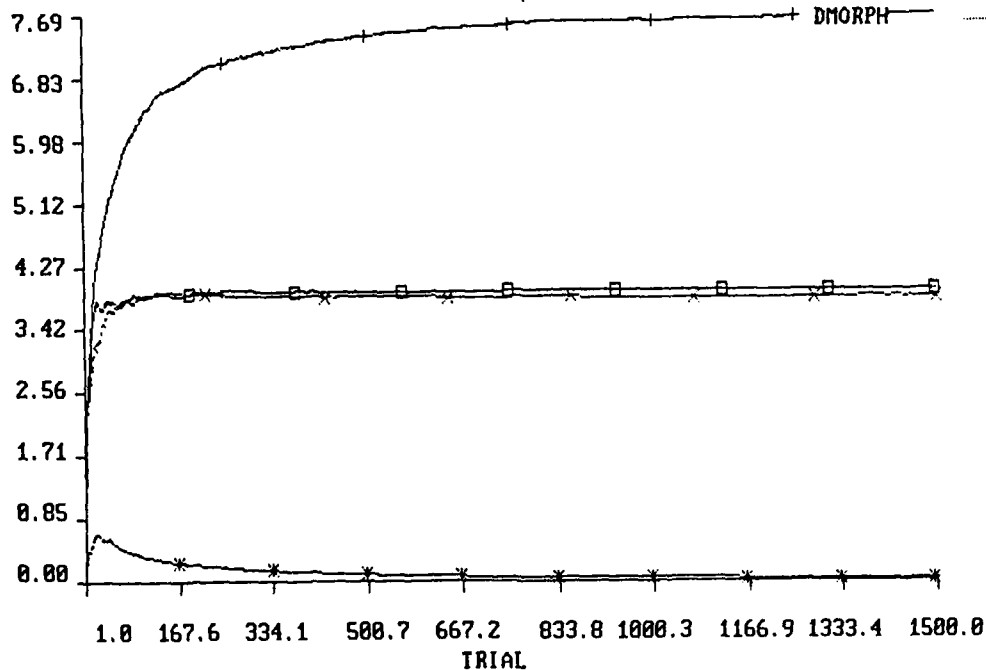
APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   6
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. Two intravariable correlations, x(1) = x(2), y(1) = y(2).

VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 2
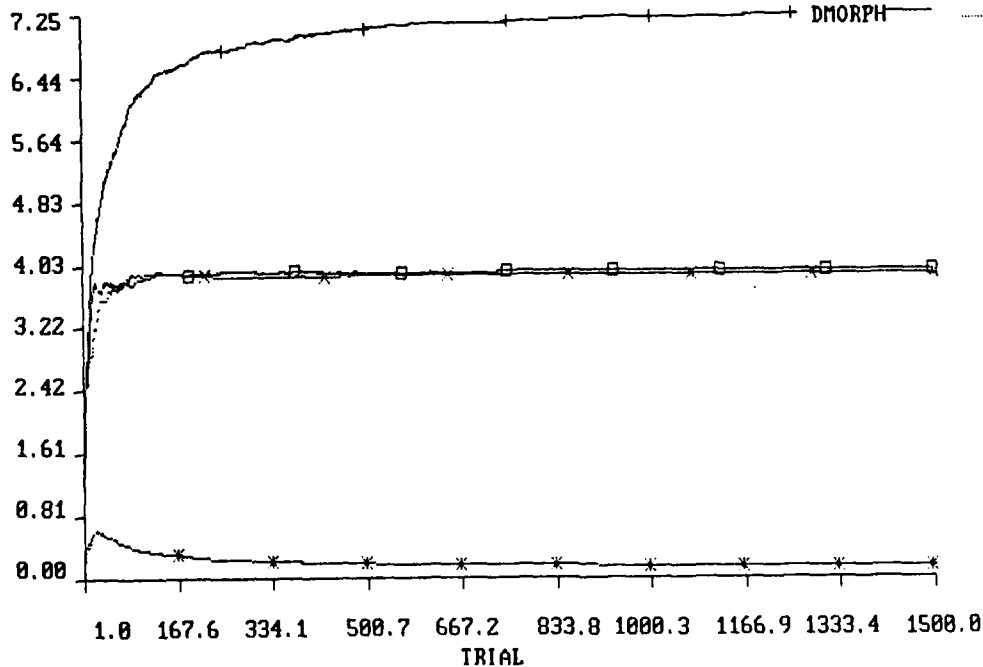
X ENTROPY: 3.8836992   Y ENTROPY: 3.9413996
DMORPH: 0.0136949   WHOLE ENTROPY: 7.7695165

COMMENTS:

FILE: EXP6.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY —+—
                                                              X ENTROPHY ·····×·
                                                              Y ENTROPHY – ⊟
                                                              DMORPH——— ······*·
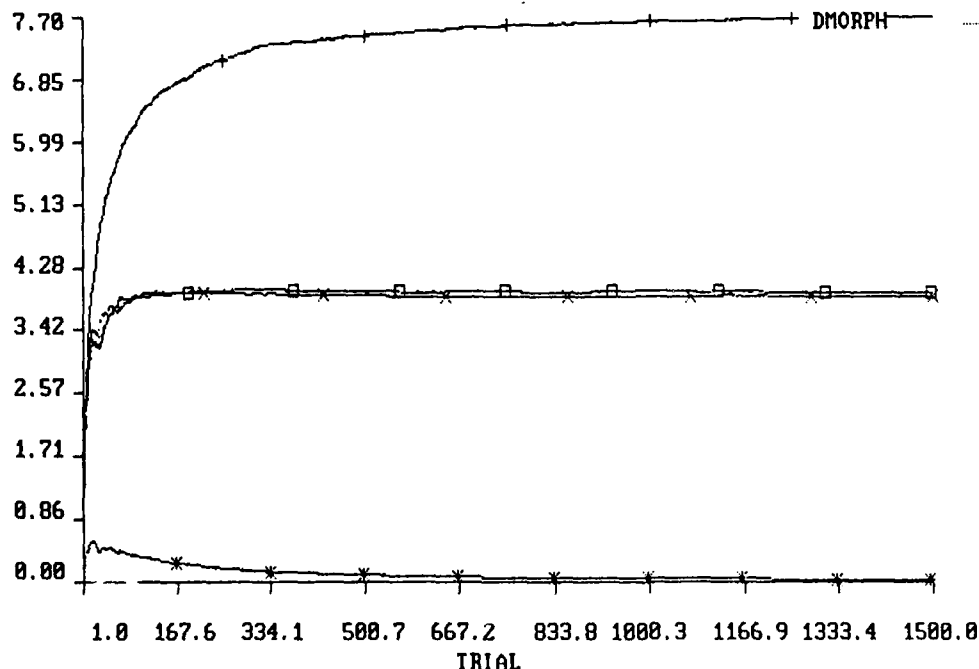
Page C.6

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney
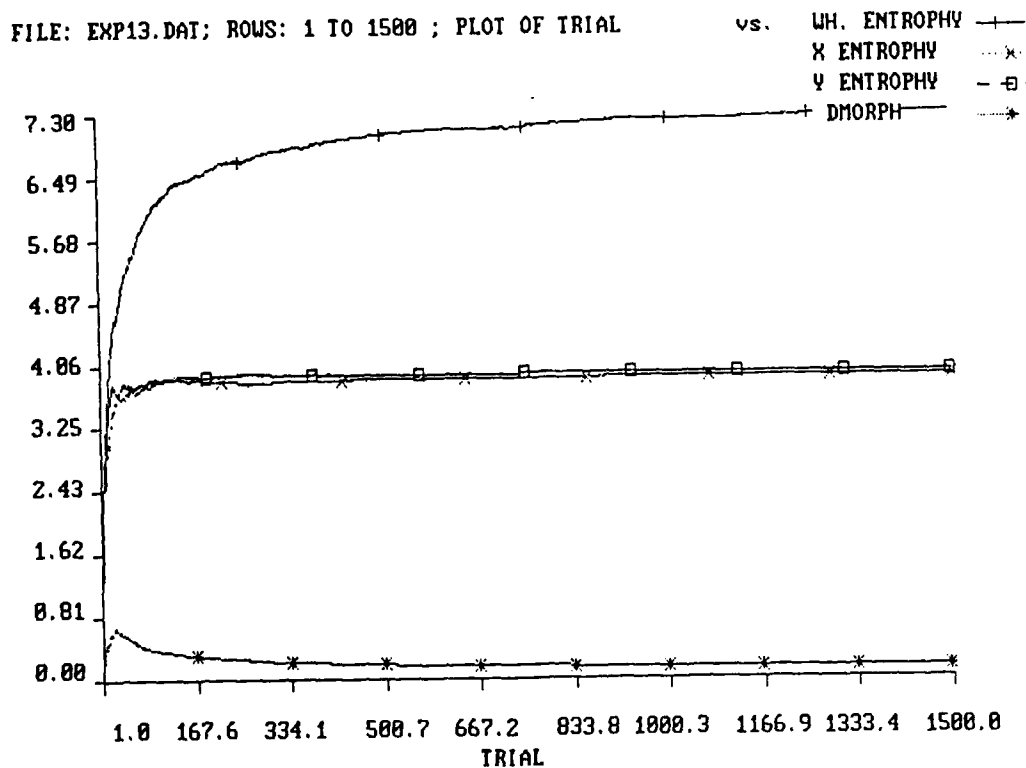
EXPERIMENT #:   7
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = y(1)$.

VARIABLES:
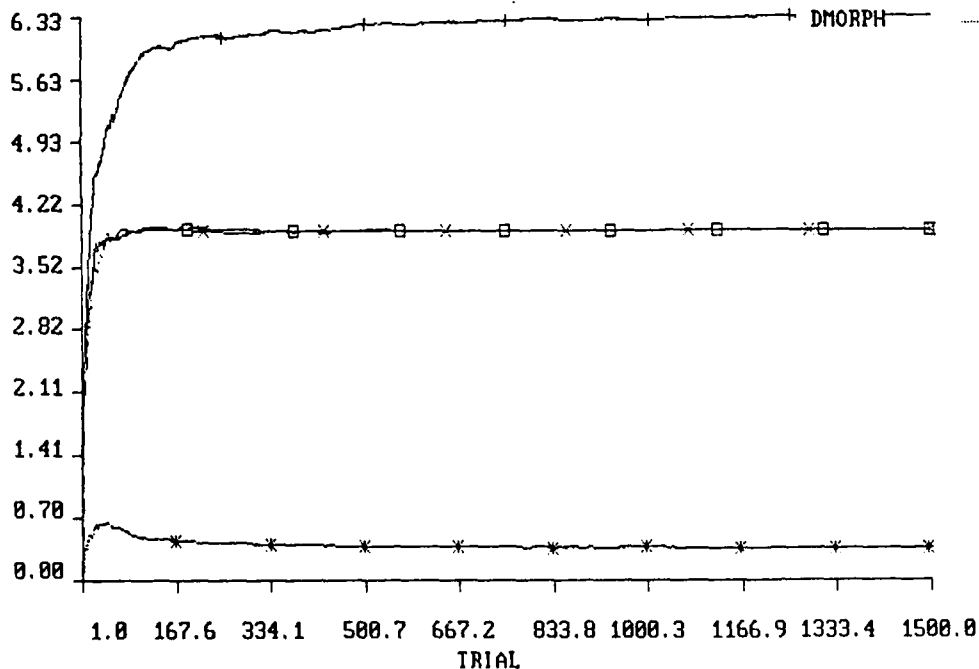
SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 3

X ENTROPY: 3.9319389   Y ENTROPY: 3.9612269
DMORPH: 0.2537242   WHOLE ENTROPY: 6.8684316

COMMENTS:

FILE: EXP7.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY —+—
                                                           X ENTROPHY   ·····x·
                                                           Y ENTROPHY   – ⊟·
                                                           DMORPH   ·······*·
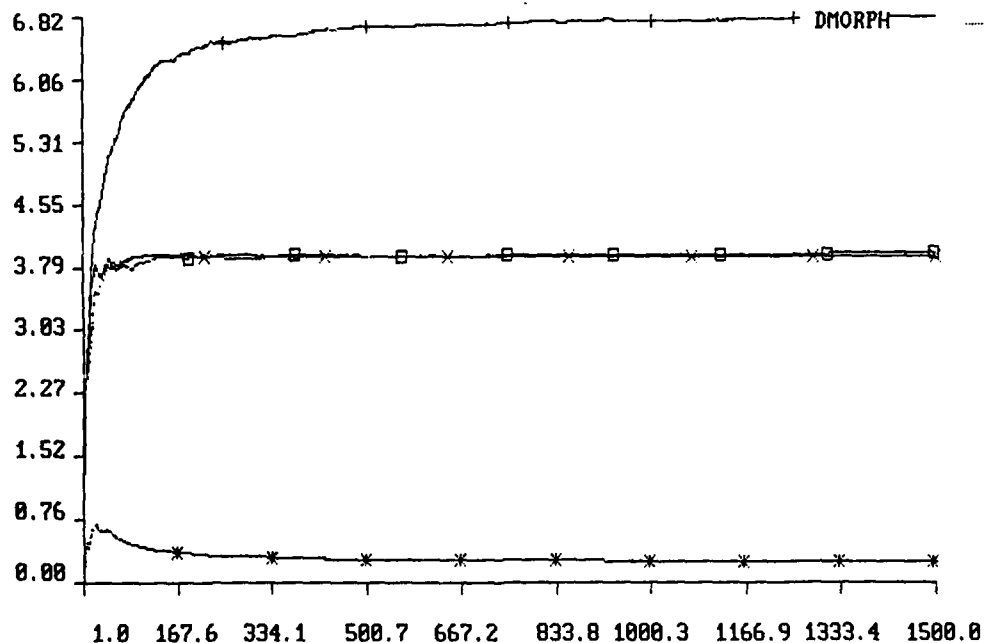
APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   8
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = -y(1)$.

VARIABLES:
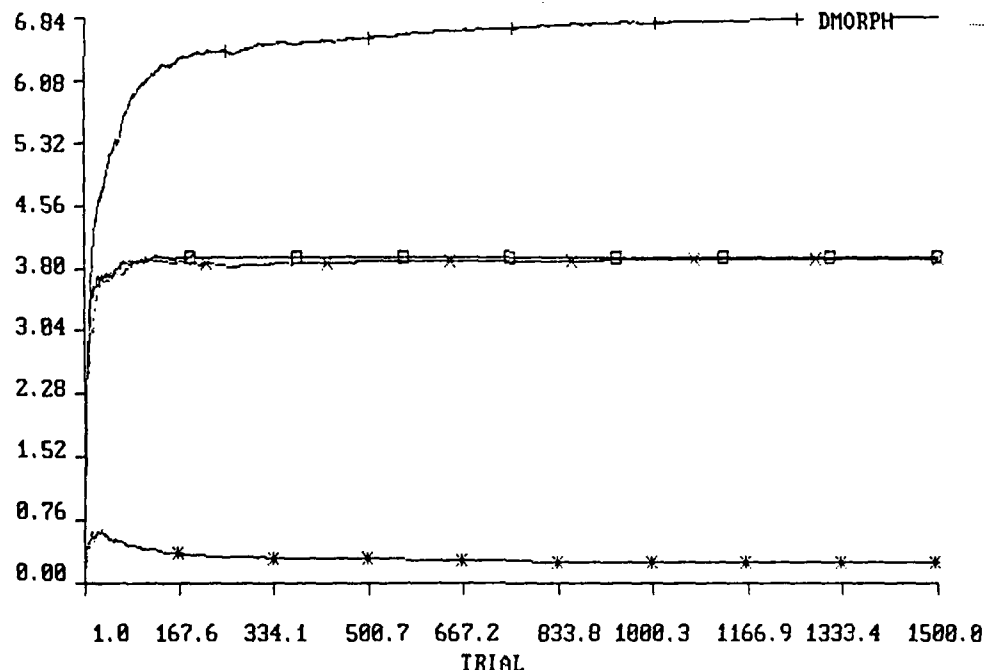
SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 4

X ENTROPY: 3.9063108   Y ENTROPY: 3.9612264
DMORPH: 0.2536734   WHOLE ENTROPY: 6.8430085
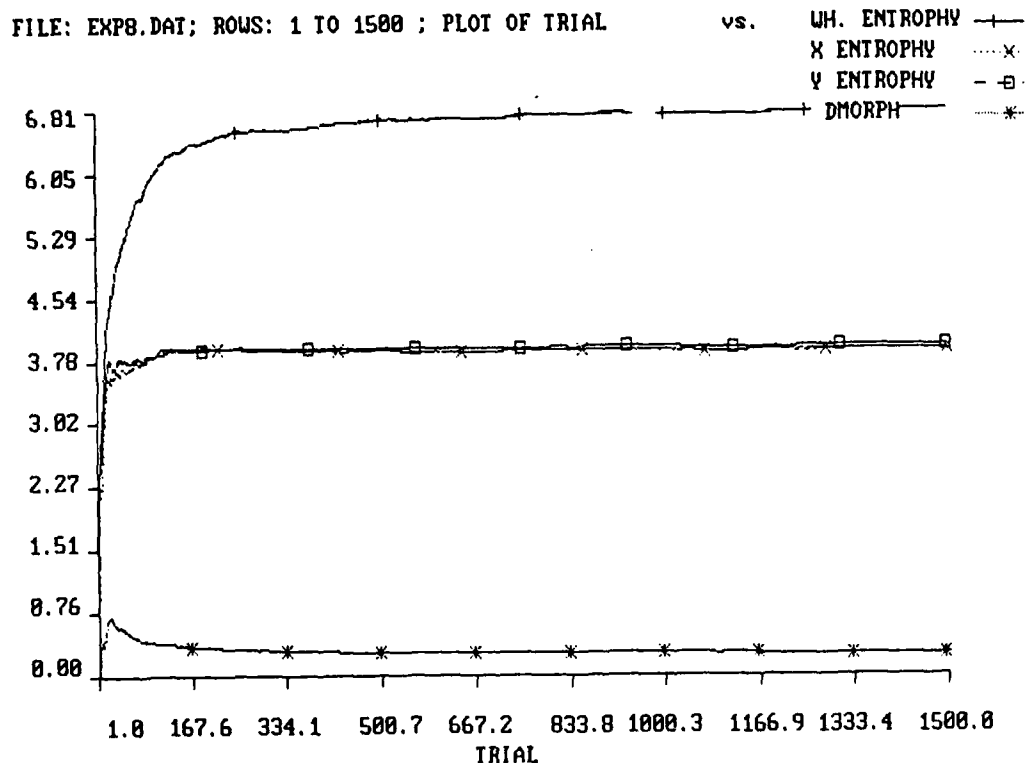
COMMENTS:

FILE: EXP8.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ——+——
                                                            X ENTROPHY   ·····×·
                                                            Y ENTROPHY   – -⊟·
                                                            DMORPH   ·····*·

Page C.8

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT:	BIOMASSCOMP
EXPERIMENT:	DMORPH Characterization
RESEARCHER:	David G. Boney
EXPERIMENT #:	9
DATE:	1/15/88

INPUT CONDITIONS: Two random variables with four components each. One intervariable correlation, $x(1) = .5 - (y(1) - .5)$.
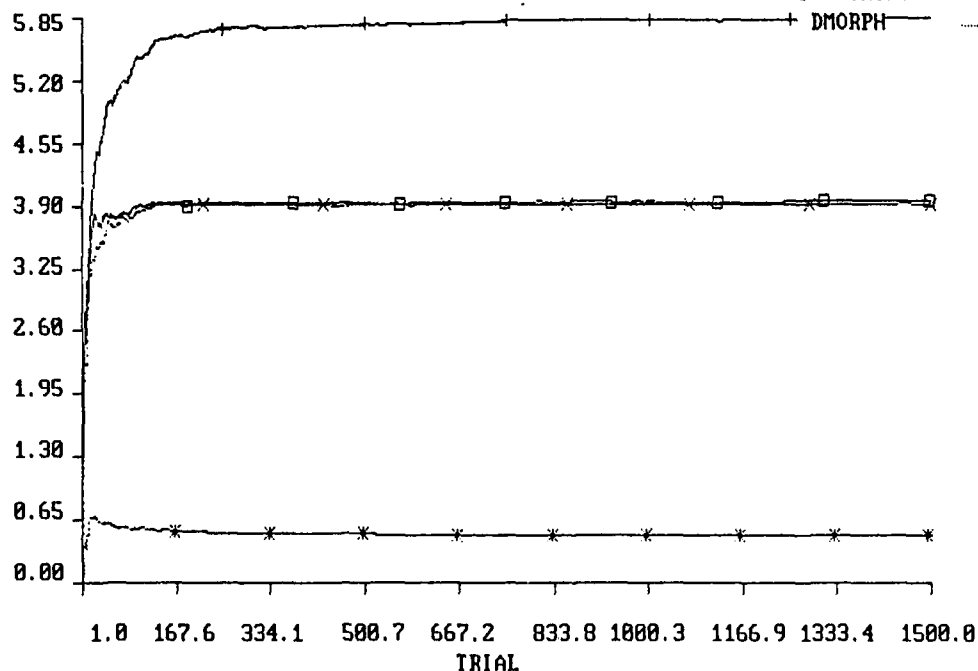
VARIABLES:

SEED:	.247E+13	LEXP:	3000			
NX:	4	NY:	4	IC:	8	
A:	0.0	B:	1.0	IFUN:	5	

X ENTROPY:	3.9063108	Y ENTROPY:	3.9612269
DMORPH:	0.2536734	WHOLE ENTROPY:	6.8430085

COMMENTS:

FILE: EXP9.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.

WH. ENTROPHY ―+―
X ENTROPHY ·····x·
Y ENTROPHY ― ⊟·
DMORPH ―――― ·····*·

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   10
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation x(1) = 10 * y(1).

VARIABLES:

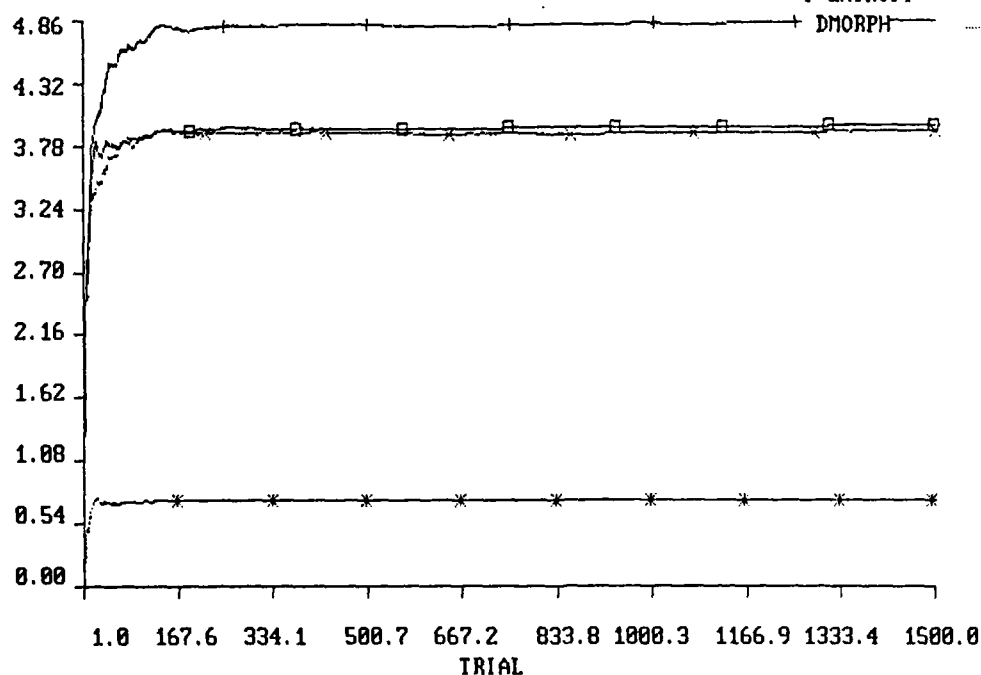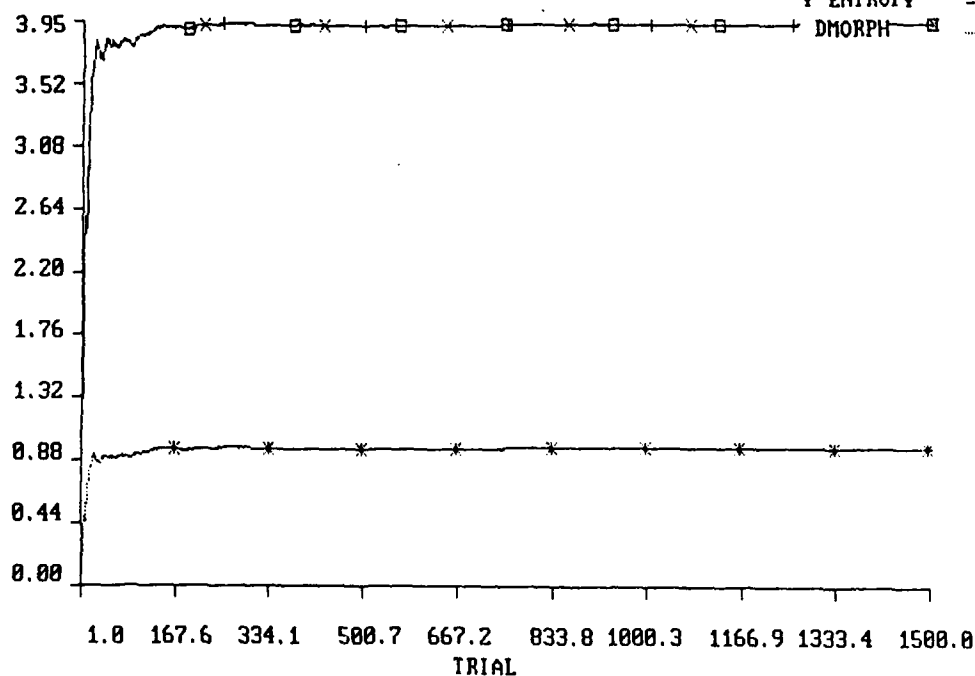SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 6

X ENTROPY: 3.9319389   Y ENTROPY: 3.9612269
DMORPH: 0.2537242   WHOLE ENTROPY: 6.8684316

COMMENTS:

FILE: EXP10.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY —+—
                                                            X ENTROPHY   ····×·
                                                            Y ENTROPHY   – ⊟·
                                                            DMORPH   ·····※·

        1.0   167.6 334.1   500.7 667.2   833.8 1000.3 1166.9 1333.4 1500.0
                                    TRIAL

Page C.10

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   11
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = .1 * y(1)$.
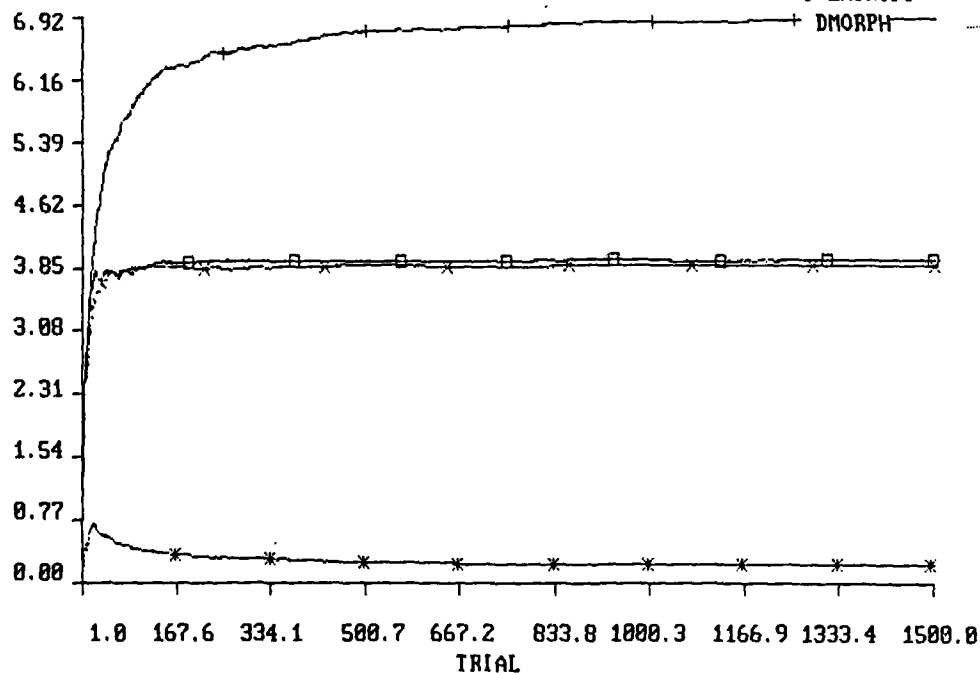
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 7

X ENTROPY: 3.9319389   Y ENTROPY: 3.9612269
DMORPH: 0.2537242   WHOLE ENTROPY: 6.8684316

COMMENTS:

FILE: EXP11.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ─+──
                                                            X ENTROPHY   ·····×·
                                                            Y ENTROPHY   ─ ─⊟
                                                            DMORPH───   ·····✳·
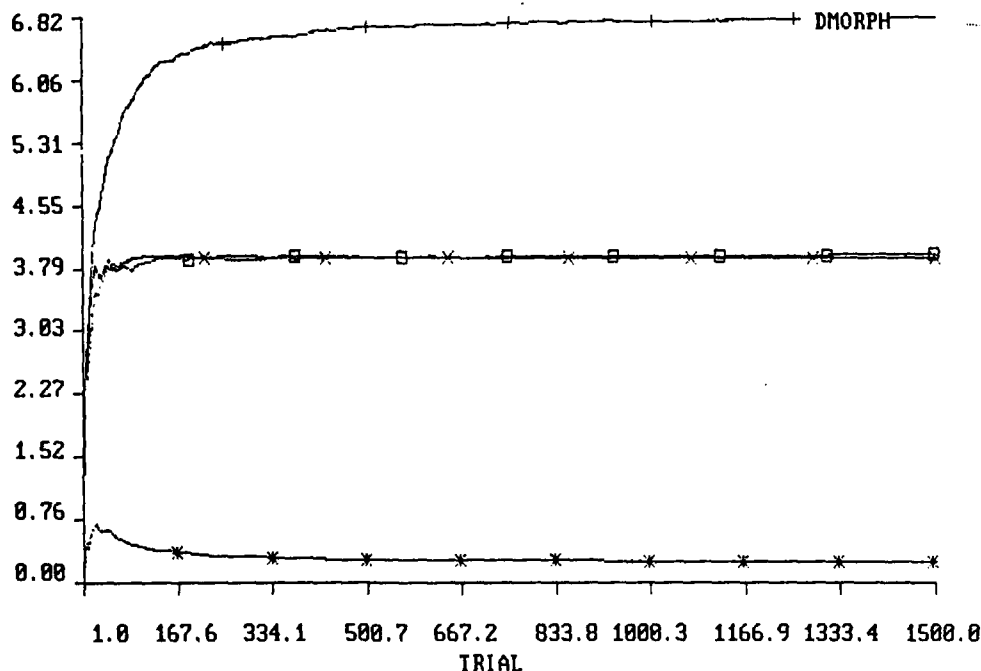
Page C.11

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   12
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = y(1) + y(2)$.
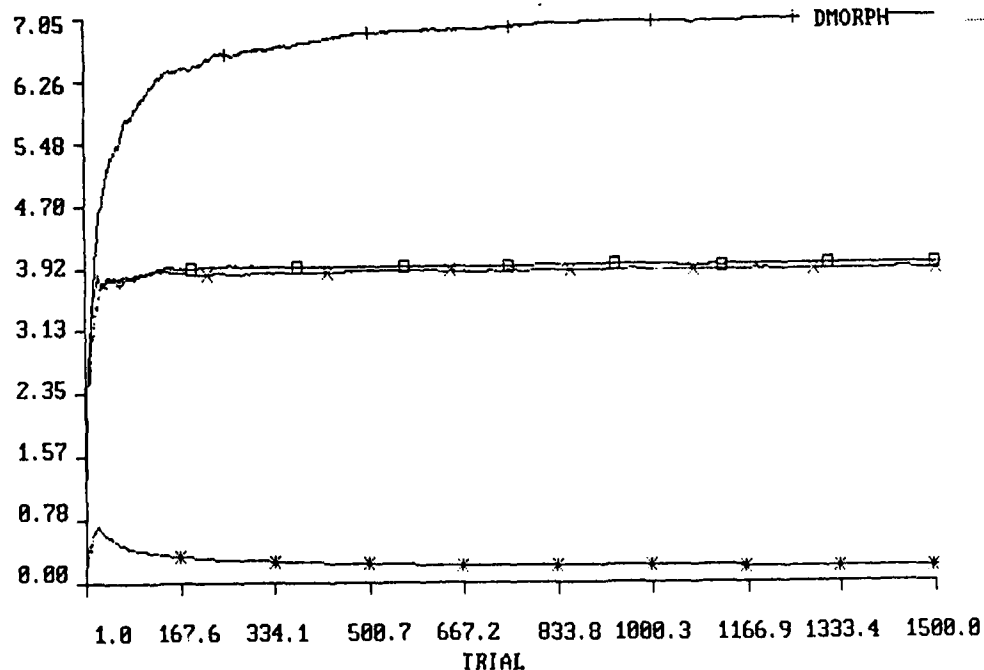
VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 8

X ENTROPY: 3.8949640   Y ENTROPY: 3.9612269
DMORPH: 0.1347252   WHOLE ENTROPY: 7.3120666

COMMENTS:

FILE: EXP12.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ─+─
                                                              X ENTROPHY   ····×·
                                                              Y ENTROPHY   ─ ⊟ ·
                                                            + DMORPH   ····*·

Page C.12

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney
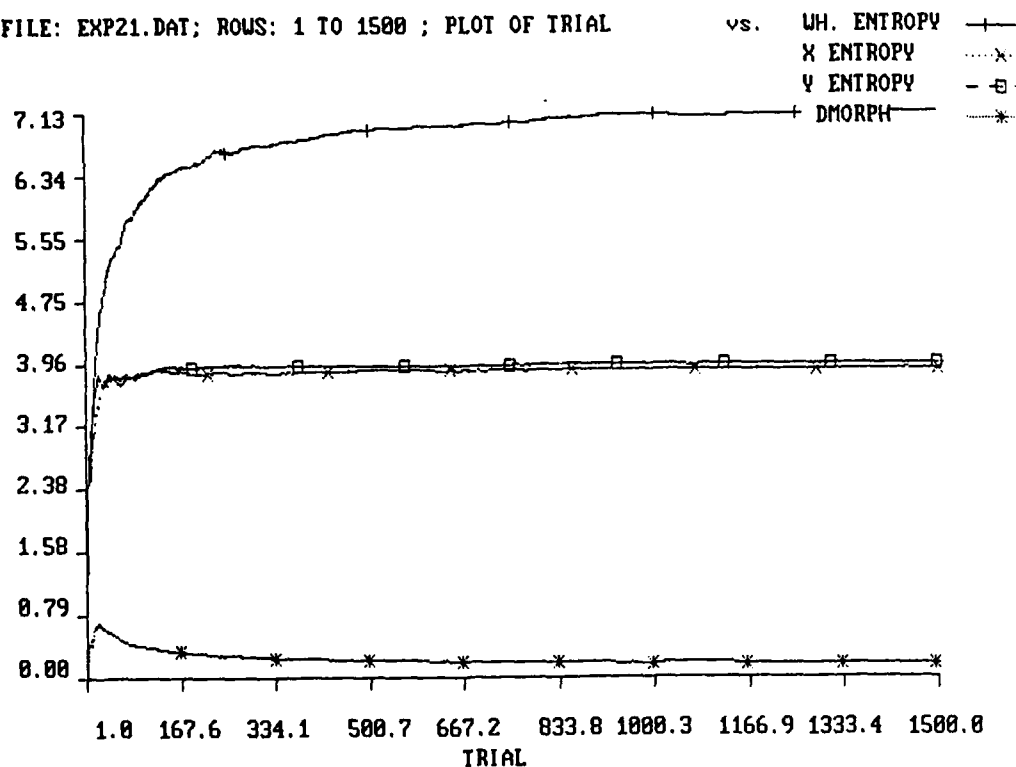
EXPERIMENT #:   13
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. One intercorrelation, $x(1) = y(1) * y(2)$.

VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 9

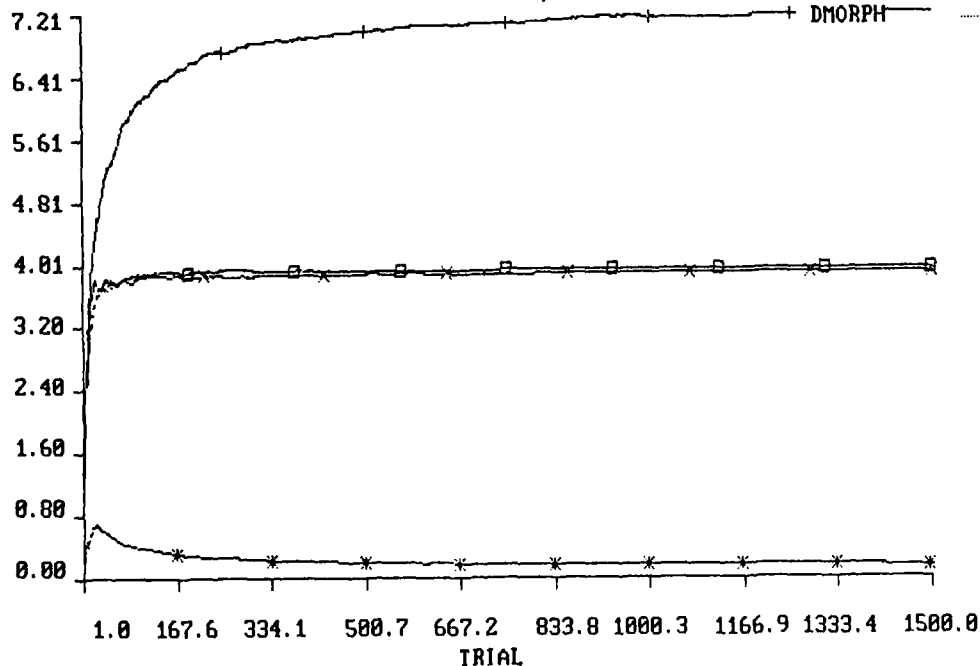X ENTROPY: 3.9062233   Y ENTROPY: 3.9612269
DMORPH: 0.122994   WHOLE ENTROPY: 7.3707037

COMMENTS:

FILE: EXP13.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ——+——
                                                              X ENTROPHY   ···×··
                                                              Y ENTROPHY   — ⊟·
                                                            —+ DMORPH——   ·····*·


```
7.30 ┤
6.49 ┤
5.68 ┤
4.87 ┤
4.06 ┤
3.25 ┤
2.43 ┤
1.62 ┤
0.81 ┤
0.00 ┤
     1.0  167.6  334.1  500.7  667.2  833.8 1000.3 1166.9 1333.4 1500.0
                                TRIAL
```

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   14
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
each. Two intervariable correlations, $x(1) = y(1) + y(2)$, $y(3) =$
$x(3) + x(4)$.
VARIABLES:
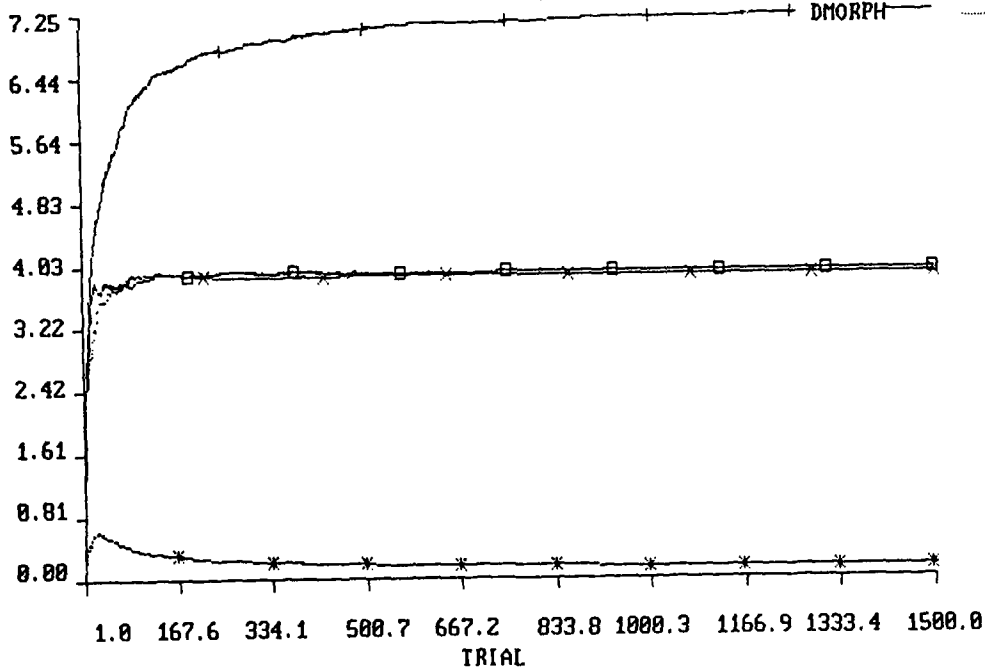
SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 10

X ENTROPY: 3.9319389   Y ENTROPY: 3.9216778
DMORPH: 0.3654028   WHOLE ENTROPY: 6.3671360

COMMENTS:

FILE: EXP14.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ──+──
                                                              X ENTROPHY   ·····x·
                                                              Y ENTROPHY   ──-e-
                                                              DMORPH   ·····*·

Page C.14

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   15
DATE: 1/15/88

INPUT CONDITIONS: Two random variables with four components
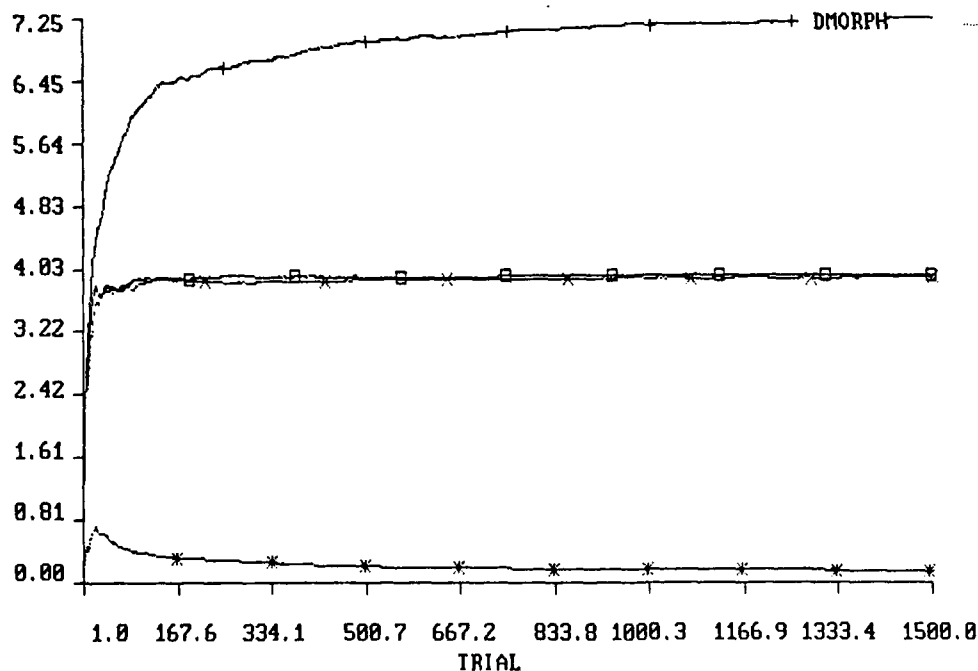each. Two intervariable correlations, $x(1) = y(1) * y(2)$, $y(3) =$
$x(3) * x(4)$.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 11

X ENTROPY: 3.9062233   Y ENTROPY: 3.9316173
DMORPH: .2310253   WHOLE ENTROPY: 6.8979411

COMMENTS:

FILE: EXP15.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPHY ─┼─
                                                              X ENTROPHY ······×·
                                                              Y ENTROPHY ─ ─⊟
                                                              DMORPH   ─────
                                                                          ·····※··

TRIAL

Page C.15

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney
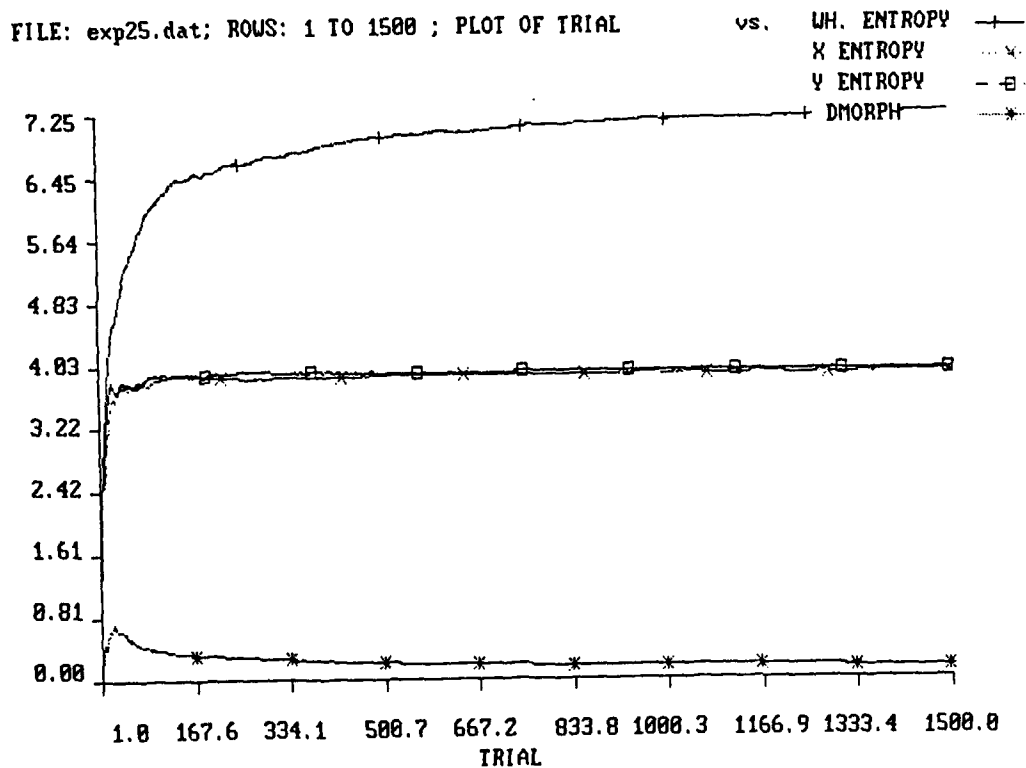
EXPERIMENT #:   16
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Two intervariable correlations, x(1) = y(1), x(2) = y(2).
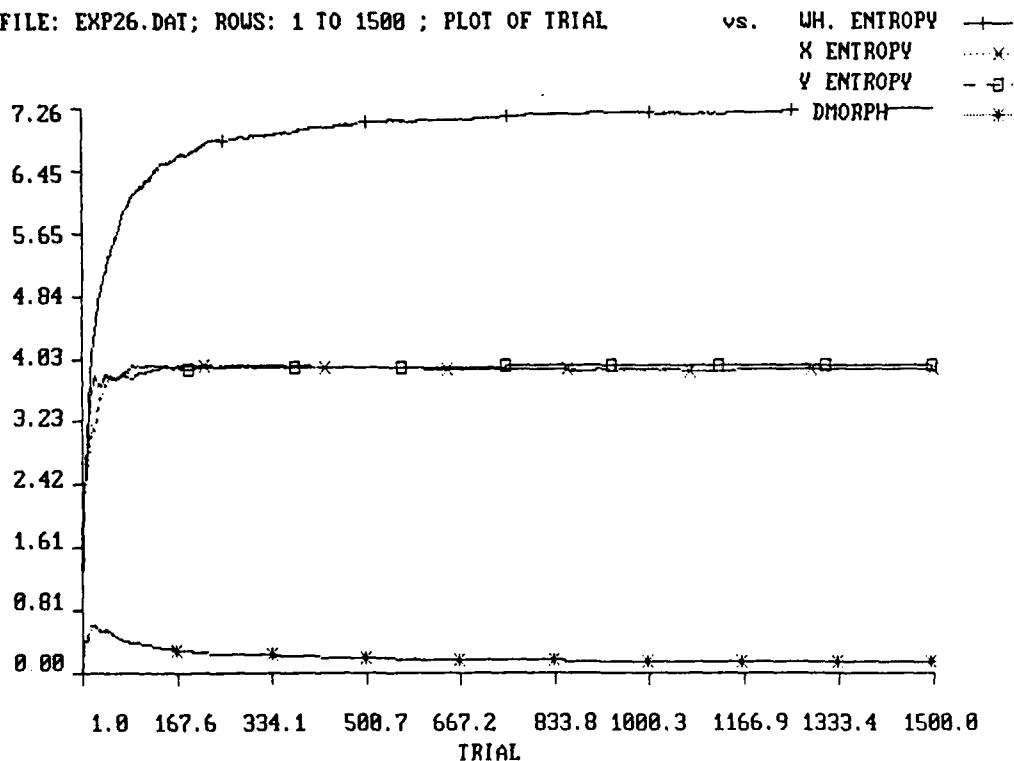VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 12

X ENTROPY: 3.9312069   Y ENTROPY: 3.9612269
DMORPH: 0.4973724   WHOLE ENTROPY: 5.8836598

COMMENTS:

FILE: EXP16.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                            X ENTROPY   ·····×·
                                                            Y ENTROPY   – ⊟ ·
                                                            DMORPH   ······*·

Page C.16

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   17
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Three intervariable correlation, x(1) = y(1), x(2) = y(2),
x(3) = y(3).
VARIABLES:

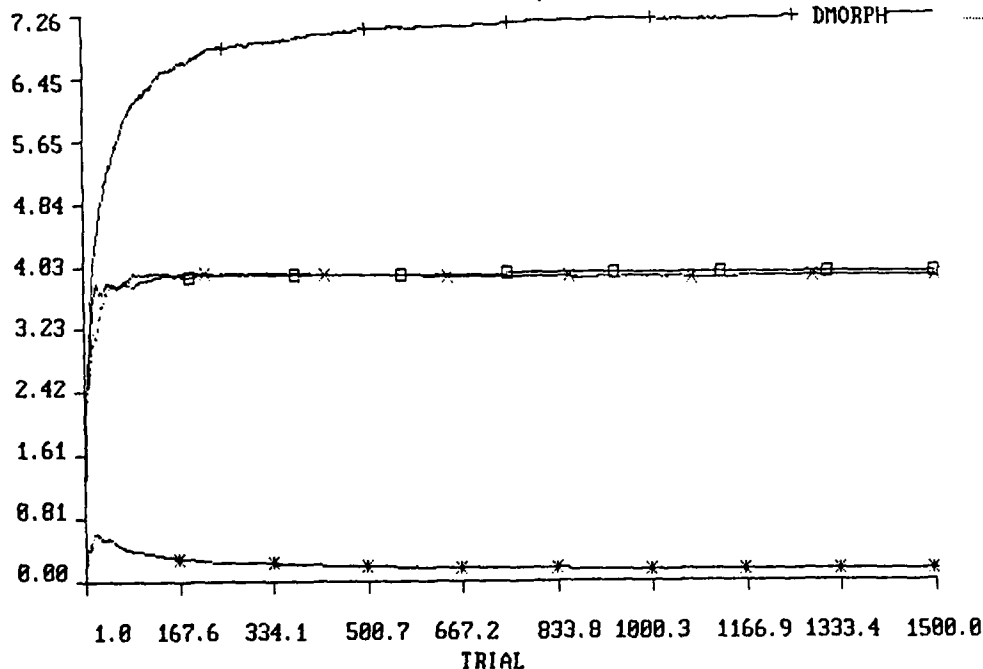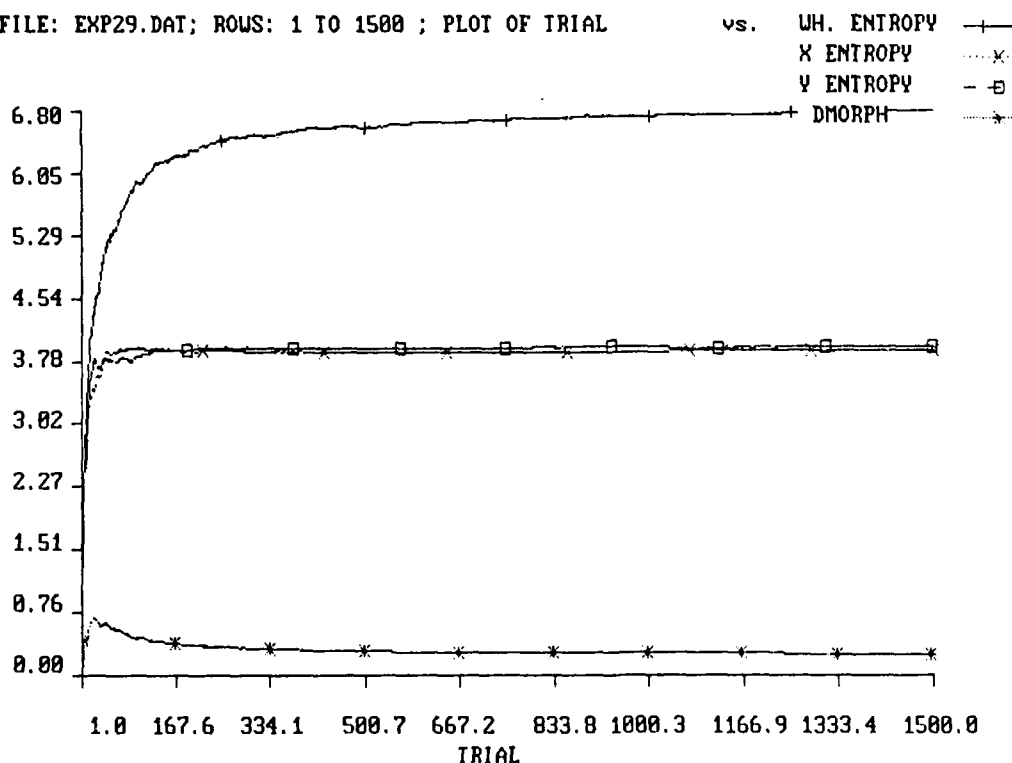SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 13

X ENTROPY: 3.8895742   Y ENTROPY: 3.9612269
DMORPH: 0.7411187   WHOLE ENTROPY: 4.8575912

COMMENTS:

FILE: EXP17.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY —+—
                                                               X ENTROPY   ·····×···
                                                               Y ENTROPY   – -θ·
                                                               DMORPH   ·····*··

Page C.17

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   18
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Four intervariable correlations, x(1) = y(1), x(2) = y(2),
x(3) = y(3), x(4) = y(4).
VARIABLES:

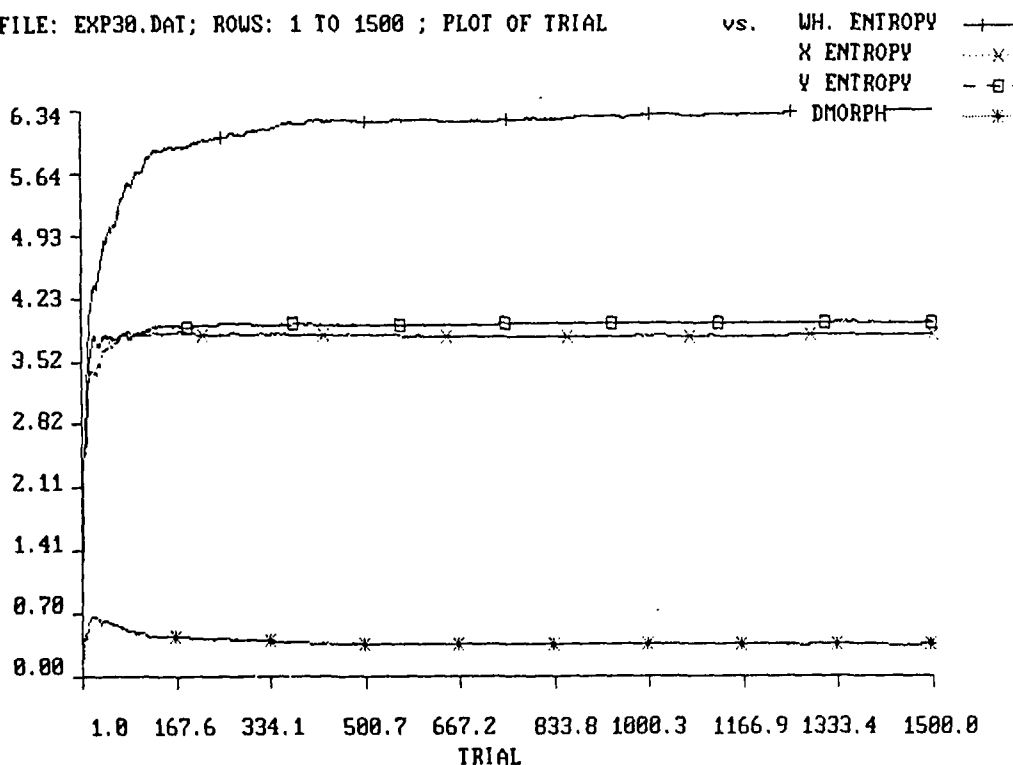SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 14

X ENTROPY: 3.9612269   Y ENTROPY: 3.9612269
DMORPH: 0.9807996   WHOLE ENTROPY: 3.9612269

COMMENTS:

FILE: EXP18.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ──┼──
                                                             X ENTROPY   ·····x·
                                                             Y ENTROPY   ─ ⊟ ·
                                                             DMORPH   ·······*·

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney
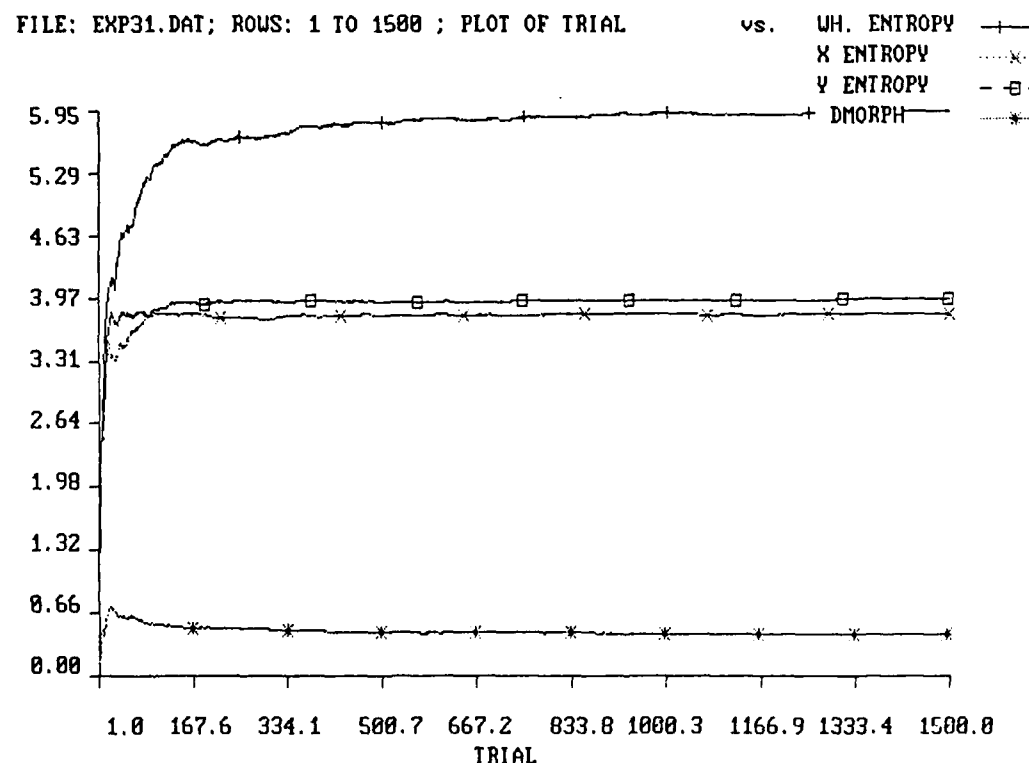
EXPERIMENT #:   19
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, x(1) = .1 * y(1) + .9 *
y(2).
VARIABLES:
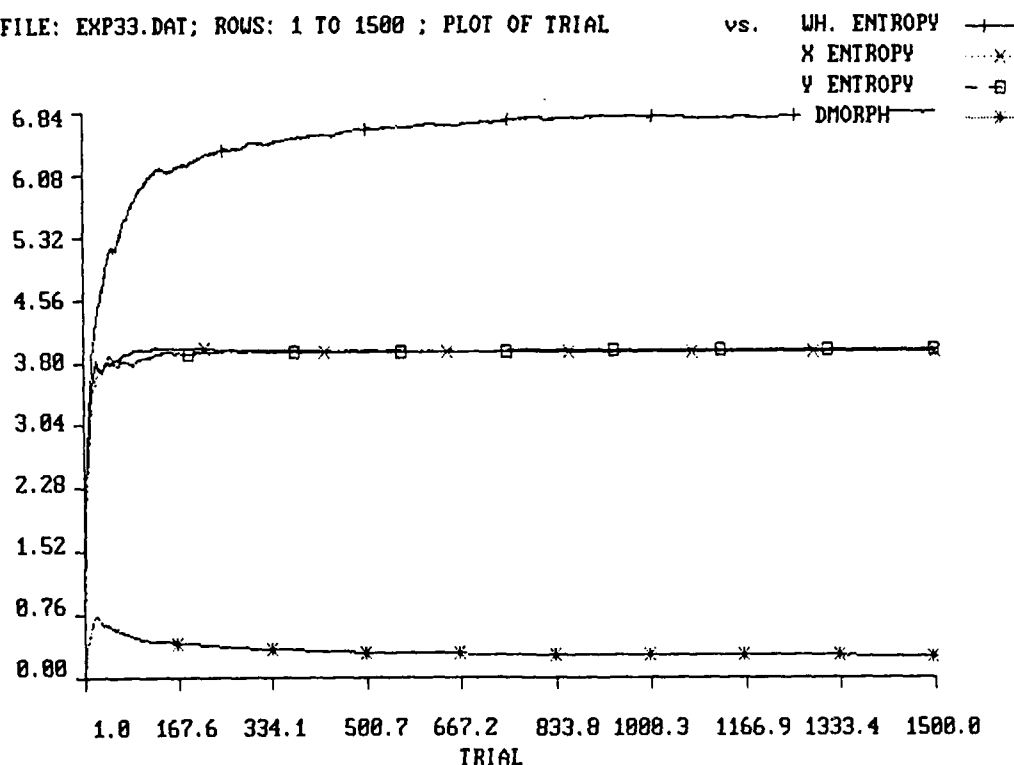
SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 15

X ENTROPY: 3.8948736   Y ENTROPY: 3.9612269
DMORPH: 0.2156875   WHOLE ENTROPY: 6.9849877

COMMENTS:

FILE: EXP19.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ──┼──
                                                            X ENTROPY   ·····×·
                                                            Y ENTROPY   ─ ⊟ ·
                                                            DMORPH   ·····*·

1.0 167.6 334.1 500.7 667.2 833.8 1000.3 1166.9 1333.4 1500.0
                              TRIAL

Page C.19

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   20
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
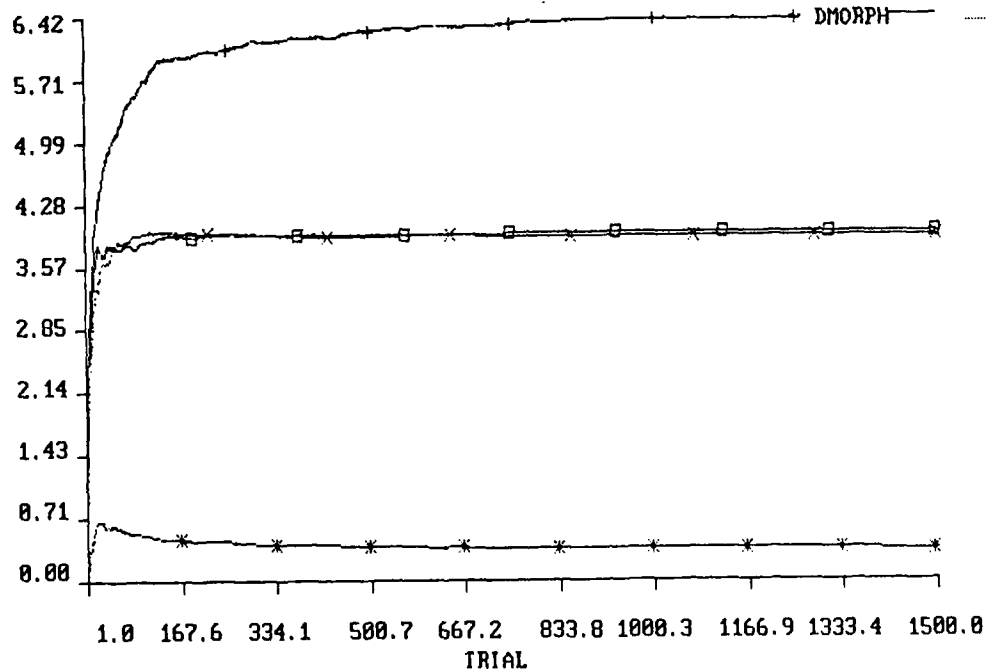each. One intervariable correlation, $x(1) = .2 * y(1) + .8 *$
$y(2)$.
VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 16

X ENTROPY: 3.8984380   Y ENTROPY: 3.9612269
DMORPH: 0.1874058   WHOLE ENTROPY: 7.1027756

COMMENTS:

FILE: EXP20.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                               X ENTROPY   ·····×·
                                                               Y ENTROPY   — ⊟
                                                           —+ DMORPH——   ·····*·

Page C.20

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

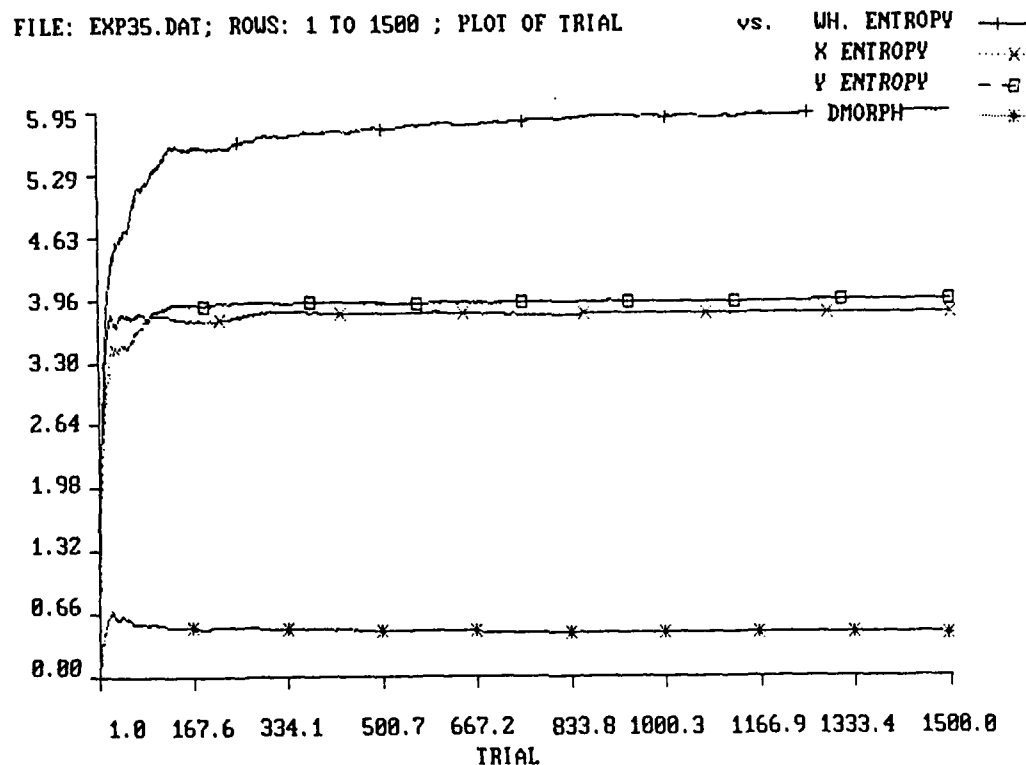EXPERIMENT #:   21
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, x(1) = .3 * y(1) + .7 *
y(2).
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 17

X ENTROPY: 3.8973515   Y ENTROPY: 3.9612269
DMORPH: 0.1651944   WHOLE ENTROPY: 7.1913958

COMMENTS:

FILE: EXP21.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                           X ENTROPY   ·····×·
                                                           Y ENTROPY   — ⊟·
                                                           DMORPH   ······*·

TRIAL

Page C.21

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   22
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, x(1) = .4 * y(1) + .6 *
y(2).
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 18

X ENTROPY: 3.8918359   Y ENTROPY: 3.9612269
DMORPH: 0.1431046   WHOLE ENTROPY: 7.2750959

COMMENTS:

FILE: EXP22.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                          X ENTROPY ·····×·
                                                          Y ENTROPY — ⊟ ·
                                                          DMORPH———— ·····*··

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   23
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, x(1) = .5 * y(1) + .5 *
y(2).
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 19

X ENTROPY: 3.8949640   Y ENTROPY: 3.9612269
DMORPH: 0.1347252   WHOLE ENTROPY: 7.3120666

COMMENTS:

FILE: EXP23.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                          X ENTROPY   ·····x·
                                                          Y ENTROPY   — ⊟
                                                          DMORPH   ·····*·

7.25
6.44
5.64
4.83
4.03
3.22
2.42
1.61
0.81
0.00

1.0 167.6 334.1 500.7 667.2 833.8 1000.3 1166.9 1333.4 1500.0
                              TRIAL

Page C.23

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   24
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = y(1) + y(2) + y(3)$.

VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 20

X ENTROPY: 3.9074244   Y ENTROPY: 3.9612269
DMORPH: 0.1411841   WHOLE ENTROPY: 7.2984409

COMMENTS:

FILE: EXF24.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY —+—
                                                           X ENTROPY   ····×·
                                                           Y ENTROPY   — ₴·
                                                           DMORPH   ······*·

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   25
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, x(1) = (y(1) + y(2) + y(3)
) / 3.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 21

X ENTROPY: 3.9074244   Y ENTROPY: 3.9612269
DMORPH: 0.1411841   WHOLE ENTROPY: 7.2984409

COMMENTS:

FILE: exp25.dat; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY —+—
                                                             X ENTROPY ···×·
                                                             Y ENTROPY – ⊟ ·
                                                             DMORPH ···*··

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   26
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = y(1) + y(2) + y(3) + y(4)$.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 22

X ENTROPY: 3.8960431   Y ENTROPY: 3.9612269
DMORPH: 0.1365768   WHOLE ENTROPY: 7.3056674

COMMENTS:

FILE: EXP26.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ─┼─
                                                             X ENTROPY   ·····×·
                                                             Y ENTROPY   ─ ⊟·
                                                             DMORPH   ·····*·

Page C.26

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   27
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. One intervariable correlation, $x(1) = (y(1) + y(2) + y(3) + y(4)) / 4$.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 23

X ENTROPY: 3.8960431   Y ENTROPY: 3.9612269
DMORPH: 0.1365768   WHOLE ENTROPY: 7.3056674

COMMENTS:

FILE: EXP27.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY —+—
                                                           X ENTROPY   ·····x·
                                                           Y ENTROPY   - ⊟·
                                                           DMORPH   ·····*·

TRIAL

Page C.27

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   29
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Two intervariable correlations, $x(1) = y(1) + y(2)$, $x(2) = y(2) + y(3)$.

VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 25

X ENTROPY: 3.8991964   Y ENTROPY: 3.9612269
DMORPH: 0.2496906   WHOLE ENTROPY: 6.8519797

COMMENTS:

FILE: EXP29.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ─┼──
                                                             X ENTROPY   ⋯⋯ӿ⋯
                                                             Y ENTROPY   ─ ⊟ ⋅
                                                             DMORPH   ⋯⋯ӿ⋯

Page C.28

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   30
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Three intervariable correlations, $x(1) = y(1) + y(2)$, $x(2)$
$= y(2) + y(3)$, $x(3) = y(3) + y(4)$.
VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 26

X ENTROPY: 3.8101032   Y ENTROPY: 3.9612269
DMORPH: 0.3473946   WHOLE ENTROPY: 6.3682823

COMMENTS:

FILE: EXP30.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                             X ENTROPY   ·····×
                                                             Y ENTROPY   — -β·
                                                             DMORPH——   ······*

Page C.29

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   31
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Four intervariable correlations, $x(1) = y(1) + y(2)$, $x(2) =$
$y(2) + y(3)$, $x(3) = y(3) + y(4)$, $x(4) = y(4) + y(1)$.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 27

X ENTROPY: 3.8055966   Y ENTROPY: 3.9612269
DMORPH: 0.4399206   WHOLE ENTROPY: 5.9900842

COMMENTS:

FILE: EXP31.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY —+—
                                                             X ENTROPY ····×·
                                                             Y ENTROPY –⊟·
                                                             DMORPH ····*·

Page C.30

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   33
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Two intervariable correlations, $x(1) = y(1) + y(2) + y(3)$,
$x(2) = y(2) + y(3) + y(4)$.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 29

X ENTROPY: 3.9173765   Y ENTROPY: 3.9612269
DMORPH: 0.2494148   WHOLE ENTROPY: 6.8712735

COMMENTS:

FILE: EXP33.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                            X ENTROPY   ·····×·
                                                            Y ENTROPY   — ⊟ ·
                                                            DMORPH———— ······*·

Page C.31

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   34
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each.

VARIABLES:

SEED: .247E+13   LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 30

X ENTROPY: 3.9007394   Y ENTROPY: 3.9612269
DMORPH: 0.3511075   WHOLE ENTROPY: 6.4439230

COMMENTS:

FILE: EXP34.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY —+—
                                                             X ENTROPY   ·····×·
                                                             Y ENTROPY   – ⊟ ·
                                                             DMORPH   ·····*·

Page C.32

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH Characterization

RESEARCHER: David G. Boney

EXPERIMENT #:   35
DATE: 1/18/88

INPUT CONDITIONS: Two random variables with four components
each. Four intervariable correlations $x(1) = y(1) + y(2) + y(3)$,
$x(2) = y(2) + y(3) + y(4)$, $x(3) = y(3) + y(4) + y(1)$, $x(4) = y(4)$
$+ y(1) + y(2)$.
VARIABLES:

SEED: .247E+13 LEXP: 3000
NX: 4   NY: 4   IC: 8
A: 0.0   B: 1.0   IFUN: 31

X ENTROPY: 3.7893291   Y ENTROPY: 3.9612269
DMORPH: 0.4477465   WHOLE ENTROPY: 5.9422097

COMMENTS:

FILE: EXP35.DAT; ROWS: 1 TO 1500 ; PLOT OF TRIAL   vs.   WH. ENTROPY ——+——
                                                             X ENTROPY   ·····×·
                                                             Y ENTROPY   – ⊟·
                                                             DMORPH——   ·······*·

Page C.33

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - BACK PROPAGATION

RESEARCHER: David G. Boney

EXPERIMENT #:   1
DATE: 2/1/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-3-4 back propagation
network that is suppose to pass through its inputs. The four
components are the following sin functions: $x(1) = .5 + .5 *$
$sin(t)$, $x(2) = .5 + .5 * sin(t - 1)$, $x(3) = .5 + .25 * sin(t)$,
$x(4) = .5 + .25 * sin(t-1)$. t is the simulation time. The y
variable is a vector of four components that is the output of the
network. The run was done with learning off and the output was
sampled at the boundry of a second.
VARIABLES:

LEXP: 1500
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1543148   Y ENTROPY: 3.5262272
DMORPH: 0.3564391   WHOLE ENTROPY: 5.0859146

COMMENTS:

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - BACK PROPAGATION

RESEARCHER: David G. Boney

EXPERIMENT #:   2
DATE: 2/1/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-3-4 back propagation
network that is suppose to pass through its inputs. The four
components are the following sin functions: $x(1) = .5 + .5 *$
$sin(t)$, $x(2) = .5 + .5 * sin(t - 1)$, $x(3) = .5 + .25 * sin(t)$,
$x(4) = .5 + .25 * sin(t-1)$. t is the simulation time. The y
variable is a vector of four components that is the output of the
network. The run was done with learning off after having learned
for 1500 seconds. Sampling was done at the second boundries.
VARIABLES:

LEXP: 1500
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1543148   Y ENTROPY: 3.1566122
DMORPH: .3363257   WHOLE ENTROPY: 4.6819711

COMMENTS:

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - BACK PROPAGATION

RESEARCHER: David G. Boney

EXPERIMENT #:   3
DATE: 2/1/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-3-4 back propagation
network that is suppose to pass through its inputs. The four
components are the following sin functions: $x(1) = .5 + .5 *$
$sin(t)$, $x(2) = .5 + .5 * sin(t - 1)$, $x(3) = .5 + .25 * sin(t)$,
$x(4) = .5 + .25 * sin(t-1)$. t is the simulation time. The y
variable is a vector of four components that is the output of the
network. This run was done with learning off and sampled once a
second at the half second boundry.
VARIABLES:

LEXP: 1500
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1540511   Y ENTROPY: 3.3335972
DMORPH: 0.4456712   WHOLE ENTROPY: 4.4079671

COMMENTS:

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - BACK PROPAGATION

RESEARCHER: David G. Boney

EXPERIMENT #:   4
DATE: 2/1/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-3-4 back propagation
network that is suppose to pass through its inputs. The four
components are the following sin functions: $x(1) = .5 + .5 *$
$sin(t)$, $x(2) = .5 + .5 * sin(t - 1)$, $x(3) = .5 + .25 * sin(t)$,
$x(4) = .5 + .25 * sin(t-1)$. t is the simulation time. The y
variable is a vector of four components that is the output of the
network. This run was done with learning off after having run for
1500 seconds with learning on. the sampling was done once a
second on the half second intervals.
VARIABLES:

LEXP: 1500
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1540511   Y ENTROPY: 3.1570110
DMORPH: 0.5151951   WHOLE ENTROPY: 3.8159778

COMMENTS:

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - KLOPF

RESEARCHER: David G. Boney

EXPERIMENT #:   1
DATE: 1/29/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-2-4 Klopf network.
The four components are the following sin functions: $x(1) = .5 +$
$.5 * sin(t)$, $x(2) = .5 + .5 * sin(t - 1)$, $x(3) = .5 + .25 *$
$sin(t)$, $x(4) = .5 + .25 * sin(t-1)$. t is the simulation time. The
y variable is a vector of four components that is the output of
the network. The inputs and outputs where in three second
intervals. This run was done with learning off
VARIABLES:

LEXP: 1475
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1604311   Y ENTROPY: 3.6817343
DMORPH: 0.4795595   WHOLE ENTROPY: 4.7713003

COMMENTS:

FILE: HK4BASE.JW9; ROWS: 1 TO 1475 ; PLOT OF TRIAL   vs. WH. ENTROPY —+—
                                                            X ENTROPY   ·····×·
                                                            Y ENTROPY   — ⊟·
                                                            DMORPH—+   ·····*·
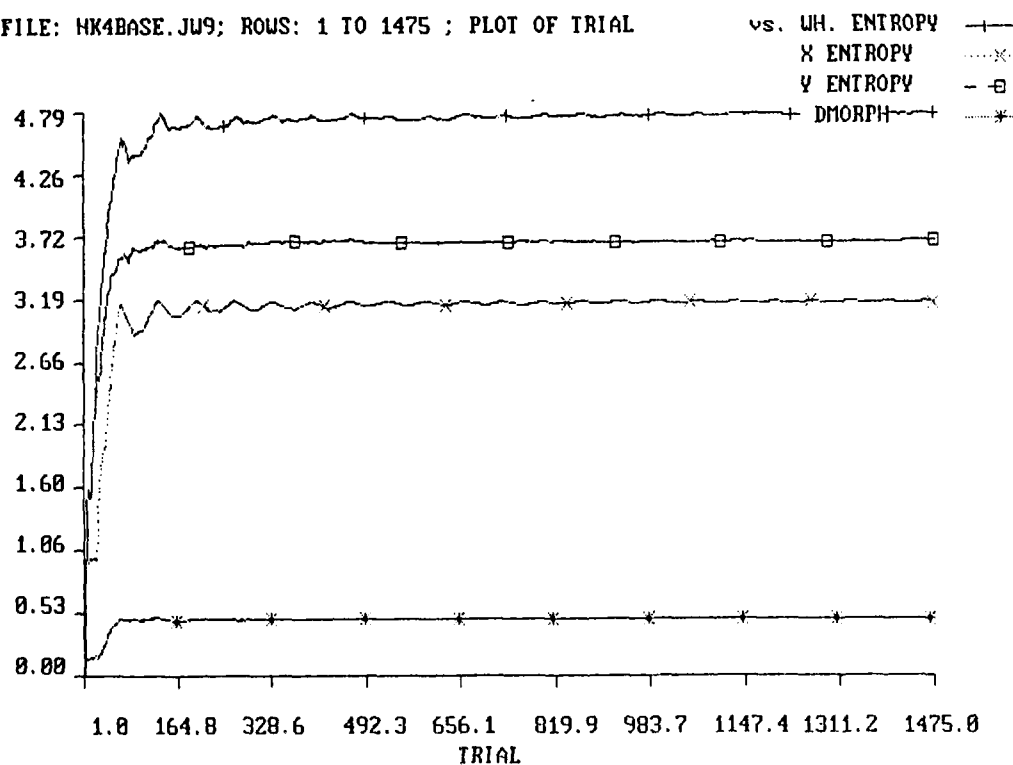
Page C.38

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - KLOPF

RESEARCHER: David G. Boney

EXPERIMENT #:   2
DATE: 1/29/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-2-4 Klopf network.
The four components are the following sin functions: $x(1) = .5 +$
$.5 * sin(t)$, $x(2) = .5 + .5 * sin(t - 1)$, $x(3) = .5 + .25 *$
$sin(t)$, $x(4) = .5 + .25 * sin(t-1)$. t is the simulation time. The
y variable is a vector of four components that is the output of
the network. The inputs and outputs where in three second
intervals. This run was done with learning off after having run
for 1500 seconds with learning on.
VARIABLES:

LEXP: 1475
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1604311   Y ENTROPY: 3.1796041
DMORPH: .4120096   WHOLE ENTROPY: 4.3539858

COMMENTS:

FILE: HK4AFTR.JW9; ROWS: 1 TO 1475 ; PLOT OF TRIAL   vs. WH. ENTROPY ——+——
                                                          X ENTROPY ······×··
                                                          Y ENTROPY – ⊟
                                                          DMORPH——+ ······*·
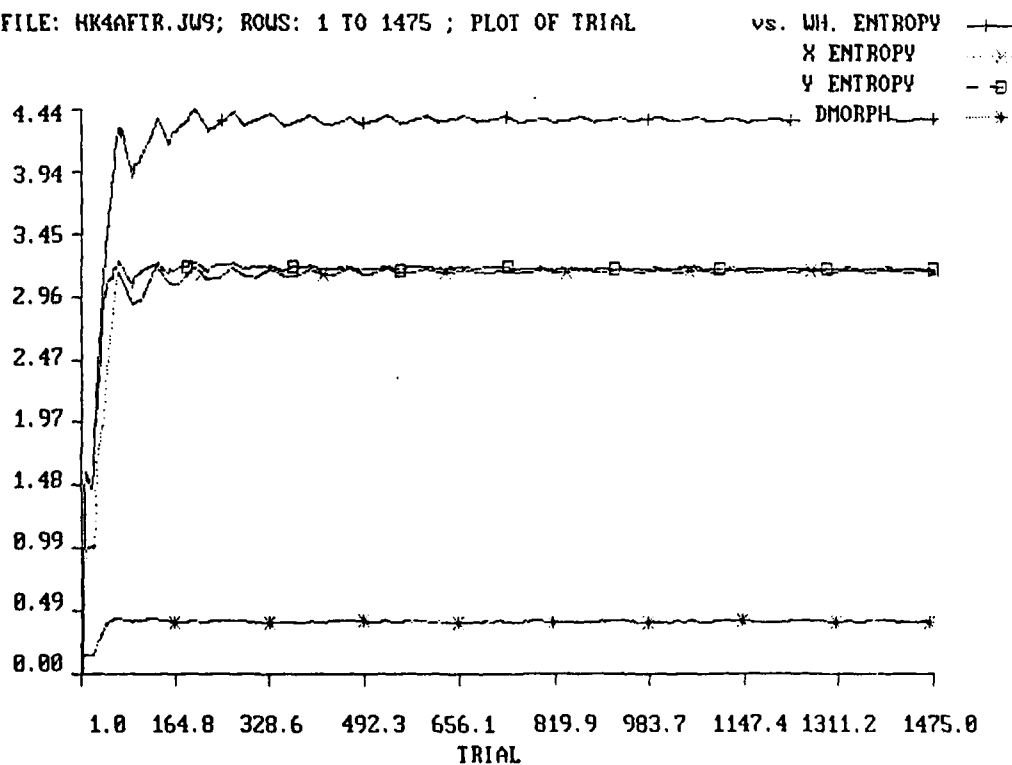
Page C.39

APPENDIX C
DMORPH EXPERIMENTS AND GRAPHS

PROJECT: BIOMASSCOMP
EXPERIMENT: DMORPH - KLOPF

RESEARCHER: David G. Boney

EXPERIMENT #:   3
DATE: 1/29/88

INPUT CONDITIONS: The x variable is a vector with four
components. This vector is the input to a 4-2-4 Klopf network.
The four components are the following sin functions: $x(1) = .5 + .5 * \sin(t)$, $x(2) = .5 + .5 * \sin(t - 1)$, $x(3) = .5 + .25 * \sin(t)$, $x(4) = .5 + .25 * \sin(t-1)$. t is the simulation time. The
y variable is a vector of four components that is the output of
the network. The inputs and outputs where in three second
intervals. This run was done after changing two of the neuron
coefficients, running the network for 1500 seconds with learning
on, and then running for 1500 seconds with learning off. The
sampling was done with learing off.
VARIABLES:

LEXP: 1475
NX: 4   NY: 4   IC: 90

X ENTROPY: 3.1604311   Y ENTROPY: 3.1176403
DMORPH: .4120096   WHOLE ENTROPY: 4.3872814

FILE: HK4AFTR2.JW9; ROWS: 1 TO 1475 ; PLOT OF TRIAL   vs.WH. ENTROPY —+—
                                                         X ENTROPY   ·····x·
                                                         Y ENTROPY   — ⊟
                                                         DMORPH——+   ·····*·

Page C.40
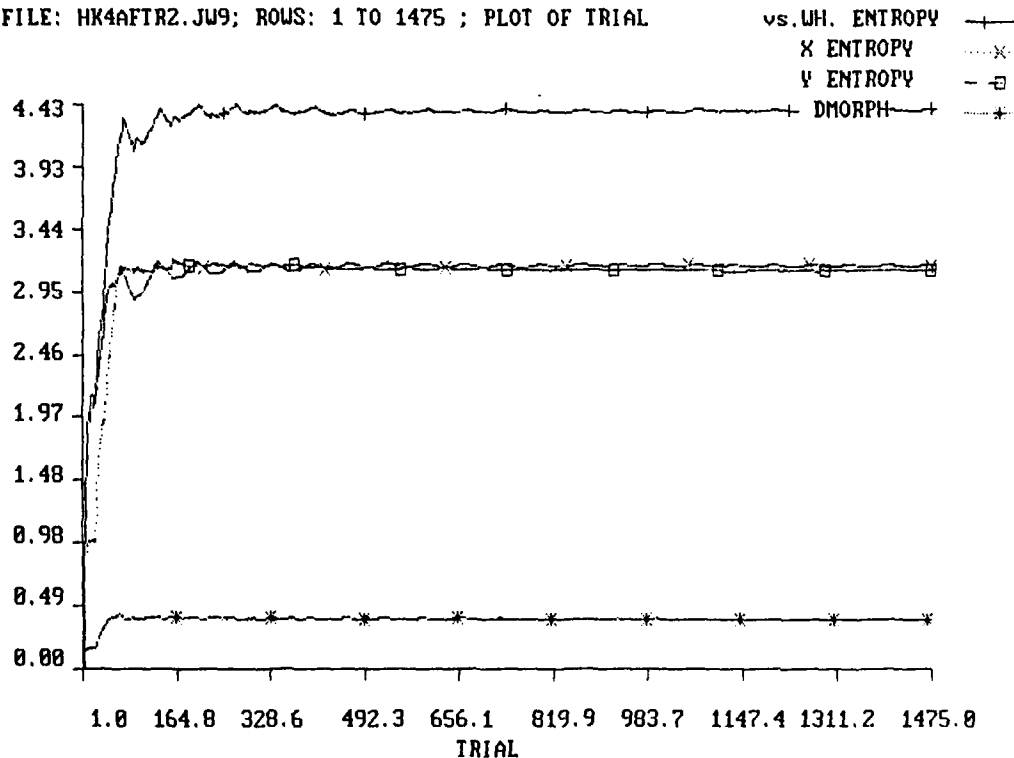
APPENDIX D

MULTIELECTRODE DATA COLLECTION ALGORITHMS

TECHNICAL REPORT

MRC-NTSU-86-1

FUNCTIONAL DESIGN OF A
REALTIME MULTISENSOR SIGNAL PROCESSING PACKAGE
FOR ACTION POTENTIALS OF NEURONS IN CULTURE

AUGUST 1, 1986

Prepared For

DEPARTMENT OF BIOSCIENCES

NORTH TEXAS STATE UNIVERSITY

By

MARTINGALE RESEARCH CORPORATION

1700 Alma Rd., Suite 305
Plano, Texas 75075-6916
(214) 422-4570

Robert L. Dawes, MRC, Principal Investigator

Joseph E. Collard, MRC

ABSTRACT

Realtime signal processing of multielectrode probes of living
neural networks is limited both by the speed and flexibility of
the host computing equipment and by the efficiency and flexibility
of the signal processing algorithm. In this report, we describe
the structural and functional design of a multichannel signal
processing algorithm which has the ability to dynamically include
or exclude processing steps and subroutines in order to maximize
the utilization of available hardware. That is, the algorithm
will perform all the processes that it can perform on realtime
data in a racetrack buffer without either overtaking the incoming
data or falling behind and being lapped thereby. Remaining
processes are performed on intermediate stored data in an off-line
(non-realtime) mode.

1.   INTRODUCTION

1.1   Review of the Problem

The primary problem addressed by this research is the fact
that it is now possible for most neurophysiology laboratories to
collect more parallel channels of data from living neuron networks
than they can afford to either save or to process in real time.
For example, Dr. Gross's MMEP apparatus is now collecting 34
parallel channels of data at 20K samples per second each. Simply
transferring that data at 2 bytes per sample to tape or disk for
subsequent non-realtime processing would fill up a 50MB volume in
a little over half a minute.

Even though a half minute's worth of data is enough to
provide useful information on the network structure, the MMEP
apparatus has the potential to not only <u>listen</u> to the culture
network, but to <u>talk</u> <u>back</u> to it as will. Without realtime
analysis of the current patterns in the network, this valuable
potential cannot be exploited. That is, while it would be
possible to "shout" at the network at random intervals and then
analyze the reactions offline later, the truly earthshaking
experiments that could be performed require the detection of
developing patterns of signals in the network (in real time) and
the selective feedback of signals to interrupt or respond to those
patterns.

With the capability for realtime interaction, it would be
possible for the first time to test certain mathematical models of
neural network behavior, such as the synaptic plasticity models
which are used to formulate explicit mechanisms for the Hebbian
learning laws (cf., Grossberg, or Hestenes [2]). In particular,
Grossberg's "outstar learning theorem" could be tested by
repeatedly injecting a pattern of signals to coincide with the
occurrence of a pattern in a naturally occurring sequence. Later,
if the artificial stimulus evokes the same response as the natural
pattern, but in the absence of the natural pattern, and in the
absence of any prior ability to evoke that response, then the
required synaptic plasticity will have been demonstrated. This
could have enormous consequences for science, with implications
not only for neurophysiology, but for psychology and computer
science as well.

To do the necessary realtime processing on which these
experiments are predicated -- indeed, to even analyze the
culture's network behavior offline in nonrealtime -- requires the
development of mathematical tools, computational algorithms, and
hardware configurations that may not now exist. Since the
hardware selection is limited more by economic considerations than
by technological capabilities, we are driven to try to coax the
resulting algorithms to be "hardware friendly". That is, we want
the algorithm to be able to adapt to the host without excessive

2

reprogramming.   The meaning of this statement will become more
clear in section 1.3 (Objectives); however, we shall now review
the existing techniques for analysis of multichannel neural
network data.

1.2   Selection of Methods

     The tasks which must be performed by the computer in order to
analyze the network data can be summarized in the following
sequence:

```
        (1)     |  Detect spikes in the signals which |
                |  are sensed in each electrode.      |
                          |
                          |
                          v
        (2)     |  Classify the detected spikes       |
                |  according to their source neurons. |
                          |
                          |
                          v
        (3)     |  Compress the data streams from     |
                |  each source neuron into a minimal  |
                |  stream still containing enough     |
                |  information to un-compress and     |
                |  recover the original data.         |
                          |
                          |
                          v
        (4)     |  Process the compressed data from   |
                |  all source neurons to identify the |
                |  communication structure of the     |
                |  total network                      |
```

The steps in this sequence become more difficult as one proceeds,
until at step (4) one finds almost nothing beyond some rather
straightforward histograms being attempted in the current litera-
ture.   Since the histograms are informative, we shall provide for
their computation, but we shall also attempt a more network-
theoretic description of the system.

1.2.1   Spike Detection

Spike detection is clearly the easiest step to perform. One
"merely" sets a threshold at a level which excludes the noise, and
whenever the (absolute value of the) digitized voltage level
exceeds the threshold, one declares that a spike has been
detected. We put "merely" in quotes because the selection of the
threshold level in a multichannel process will be done by the
computer according to some algorithm, and this algorithm is
considerably less transparent than the thresholding instruction.

Establishing the threshold -- an easy task for a person
looking at an oscilloscope trace -- is essentially a problem in
"constant false-alarm rate" (CFAR) methods. Digital CFAR thresh-
olding is discussed by Rohling in [5]. In the absence of spiking
signals, i.e., when only noise is being sensed, the threshold can
be determined by computing a histogram of the digitized voltages
and finding a level within which enough of the samples lie, so
that only once in a specified number of seconds does a noise
sample lie outside the threshold.

Unfortunately we usually have to take the noise as it comes:
with some signal added to it. Therefore in order to set the CFAR
threshold we have to mask out the spikes so that they are excluded
from the histogram. Of course, we can't use thresholding to find
the spikes, because it is the threshold we are trying to find! So
we have to use some other a-priori knowledge to find and mask the
spikes. If the computational application of this knowledge were
quicker than thresholding, then we would naturally use it for the
realtime spike detection; but it isn't, so we apply it prior to
realtime processing in a so-called "learning" mode.

There are a number of different masking techniques, depending
on the application, but one that comes to mind exploits the
continuity (smoothness) of the spike trace as opposed to the
roughness of the noise. Thus the algorithm will examine a number
of consecutive samples to see if their magnitudes are all
unusually large (compared to an unmasked histogram) and all on the
same side of the origin. If so, an appropriate amount of data
around the suspected spike is masked off, i.e., removed from the
histogram. This process continues to be performed on the learning
data set until no further spikes can be identified, whereupon the
remaining histogram represents an approximation of the probability
density function of the noise alone. The threshold can then be
selected at that value for which the number, n, of samples whose
magnitudes exceed the threshold, divided by the length, T, of the
learning sample in seconds is most nearly equal to the desired
false alarm rate.

One could detect spikes by algorithms that are more
complicated than simple thresholding, such as by matched
filtering, maximum likelihood estimation, and many others (each
requiring its own "learning mode" and each also requiring its own
thresholding operation), but these methods can be reserved for

4

some post-detection processing in the remaining steps of the
sequence.

1.2.2 Spike Classification

     The next task is to determine whether the signal on a given
electrode is the superposition of spike trains from more than one
neuron source, and if so, to separate the signal into its
constituent parts. A survey of computer methods for this
separation task was given by Schmidt [6] in 1984. Other
techniques can also be found (cf, Okada and Maruyama [4]). Our
approach in this section is to select a sequence of methods based
on a progression from the computationally simple to the more
difficult. We choose this approach because the structural design
of our algorithm (Chapter 2) calls for the use of simple methods
on channels where simple methods suffice, and harder methods where
they are required, thus using available processing time most
efficiently. (Hardware configurations employing a dedicated
signal processor in each channel will not need to avail themselves
 of this choice.)

     As with the detection process, spike classification is done
in two parts. During a learning mode each channel is evaluated to
determine the values of some variables which will be used in the
real time mode and which are expected to change very slowly, if at
all. For spike separation, these variables will partition the
channels into subsets, each of which can be de-interleaved by a
different class of algorithm.

     The first of these variables will identify the number of
sources being received on the channel. If only one source is
being received, then step (2) of the sequence is trivial. The
second variable will be a vector whose components identify the
amplitudes at which distinct sources are found and the number of
distinct sources that are found at each amplitude. If no
amplitude bin has more than one source contributing to it, then
the classification can be performed by amplitude discrimination.
But if any bin has more than one source then some form of waveform
discrimination will have to be used to separate the sources.

     The easiest way to determine that there are more than one
source neuron represented in the signal is to perform a one-
dimensional cluster analysis on the peak voltage in each detected
spike. (A general description of cluster analysis algorithms is
given in Appendix A.) If more than one cluster is found, then
there are more than one source. The fact that the converse of
that statement is false necessitates the use of more complicated
algorithms for the classification process, but since these more
complicated mthods work best when the peaks are presented in
constant-amplitude clusters, we may as well do the easiest job
first.

To do the amplitude (peak) cluster analysis, one must first
detect and measure the height of the greatest local extremum
within a single pulse-width of the threshold crossing (detected
in step 1).   this can generally be done without any multiplies or
divides.   (See Appendix B, or reference [4]).   Using a cluster
diameter that is large enough to allow for known amplitude
variations from single sources, one then sorts the peak values
into a histogram (see Tou & Gonzalez [6]) to find the clusters.

To determine whether the peaks assigned to a given cluster
break down further into different wave shapes one can then do an
n-dimensional vector cluster analysis, where n is the number of
samples after a threshold crossing needed to cover all spike
waveforms.   The n-dimensional clusters are found analogously to
the way the amplitude clusters are found, but the distance measure
is a little more involved and the cluster diameter is more
difficult to establish (See Appendix A).

Once the waveform clusters are identified, it might be
possible to do the realtime assignment of spikes to the
appropriate cluster with an algorithm that is simpler than a
template comparison or a matched filter.   But even with a
"hardware friendly" algorithm, it is fair to assume that there is
a vector or array processor lurking in a wait-state nearby,
eagerly contemplating its next victim.   Therefore, we shall prefer
to simply vectorize the distance between the realtime peak and the
centers of the clusters to assign it to its source.

1.2.3   DATA COMPRESSION

The first step in data compression is almost taken care of
in tasks (1) and (2) simply by detecting and classifying the
spikes in each channel.   By delivering a report of the TIME when
the spike was detected over the threshold the SOURCE which emitted
it, and (perhaps) the measured spike amplitude, one has compressed
the 20 or so sample values representing the spike, and the 80 or
so preceding sample values representing noise alone into only one
or two values from which a replica of the spike (sans noise) can
be reproduced.

One essential item in the description of the SOURCE (though
it may actually be irrelevant as far as the neuronal "message" is
concerned) is the wave shape of the spike, which was discovered in
the classification step.   Since this shape is expected to change
only very slowly, if at all, during the experiment, it can be
identified (via pointers or links to a template library) with the
array into which the TIME values are reported.   Thus, if one wants
to resurrect a replica of the raw data which was recorded from a
particular source neuron, one can retrieve the sequence of times-
of-arrival of spikes from the array associated with that source,
and at those times construct a pulse with the shape in the
template library that is linked to the source.   If amplitudes are
considered important, the (normalized) templates can be scaled by

6

the amplitude that was stored alongside the time of arrival.

Data compression is possible whenever there are replicated patterns in the data that can be parametrized and reduced to a symbol, followed by a list of values of the parameters. Thus, in the previous paragraph, raw voltage samples from an electrode which is sensing two sources is compressed into the following symbols and parameter lists:

SOURCE1((T11,A11),(T12,A12), ...)

SOURCE2((T21,A21),(T22,A22), ...),

where T_{ij} is the time of arrival of the j-th pulse from the i-th source, and A_{ij} is its amplitude. The name, SOURCE1, is taken to be equivalent to the unchanging characteristics of the source. Similarly, mathematicians use symbols like "SIN", and "LOG" to compress the descriptions of families of functions, and by supplying parameters like the frequency of the sinusoid, and the base of the logarithm, they can then resurrect a graphical representation of the function.

"Bursting" is an observable feature of signals drawn from certain kinds of neurons. Even though the literature shows little agreement on a definition of what might constitute a burst, we can sidestep that issue for the purpose of data compression. For our purposes, it is sufficient to establish a library of certain patterns occurring in the compressed spike reports, and when those patterns are detected, reduce them to a more compact representation. We suggest the following definition:

A "burst" is either (1) three or more consecutive spikes, whose amplitude sequence lies within a martingale envelope rooted on the first spike, and whose interval sequence lies within a martingale envelope rooted on the first interval; or (2) any single spike which fails to fit in a sequence of the previous category. The envelope to be used on the amplitude sequence should be of the form,

$$a + (b-a)\exp(-kt) +/- vt ,$$

where the values of a, b and k can be determined from the first three amplitudes in the candidate sequence (since vt is nearly zero at the start), and v is the variance of the amplitude process. Similarly, the envelope to be used on the interval sequence should be of the form,

$$b' - mt +/- v't ,$$

where b' and m can be determined from the first two intervals, and v' is the variance of the interval process.

Thus, in a burst, one expects the amplitudes to fall off along some declining exponential starting at the first pulse, plus

or minus a margin that gets a little wider as the burst proceeds;
and one expects the interpulse spacing to increase approximately
linearly with time, again with a margin that increases down the
line.   The parameters a, b, b', k, and m are measured in a learn
mode, and taken to be characteristic of the source; while v and
v' are taken to be rejection thresholds outside of which the
candidate burst sequence terminates.

1.2.4   NETWORK ANALYSIS

     The state of the art in network communication analysis of
living neuron networks is still rather primitive, which is to be
expected due to the quite recent emergence of the technology for
simultaneous sensing of numerous points within isolated networks.
The principal tool for the analysis is the pairwise correlation of
features of the spike trains by way of the cross-correlation
histogram.   These methods are described in Chapter 10 of MacGregor
and Lewis [3], and in several other papers (e.g., [1]).

     We feel that although these correlograms are useful tools for
sparsely connected networks such as might be found in aplesia or
in sensory ganglia, we are not likely to find statistically
significant correlations appearing in the more densely connected
networks.   The reasoning here is that networks such as found in
the mammalian cortex are structured for the efficient sorting of
coordinated patterns of input signals, rather than for serving as
in-line amplifiers of single inputs.   Consequently, it is only
when a synaptic input is a part of a coordinated pattern of inputs
that it will participate in the generation of spiking or bursting
at the output of the afferent neuron.

     In order that these experiments with the MMEP apparatus
should fulfill their potential, we feel that they should result
not in the publication of a report that is full of histograms and
other statistical humdrum, but rather that they should be used to
confirm or eliminate specific quantifiable hypotheses regarding
the possible mechanisms of learning, recall, synaptic plasticity,
memory storage, and the like.   To do this requires the use of
models which link the hypotheses to certain parameters which are
susceptible to measurement with the apparatus.

     One such model is provided in the system of coupled nonlinear
ordinary differential equations known as Grossberg's Field
Equations (nicely presented by Hestenes in [2]).   These equations
describe the incremental effect on the ionic potential energy of
certain pulse patterns arriving at the synapses.   The details are
important, but they can be summarized by pointing out that for the
excitatory synapses, the effect of the incoming signal in driving
the neuron toward its firing threshold is proportional to the
recent history of the pulse repetition frequency (PRF) on that
synapse.   The constant of proportionality is a characteristic of
the synapse that can be modified by the coincidence of neuronic
firing and input to the synapse.   It is called the synaptic

coupling coefficient.   The amount of history that is relevant is controlled by the rate at which the neuron will dissipate its energy without firing.

According to this model, one would expect to find correlations between the onset of bursts from a given neuron and the PRF history on its synapses.   This suggests that converting the spike data to the corresponding PRF history, and "stacking" these histories whenever a burst occurs at the output of a particular neuron, might accumulate strong peaks on signals that are connected to excitatory synapses.   On the other hand, if the stacks are triggered at the onset of blank intervals in the neuron's output, then the presence of strong peaks would indicate an inhibitory influence.   In contrast to the cross-correlation (interval) histogram, which attaches significance to the influence of a single spike at the input to a subsequent single spike at the output, this technique attaches significance to the recent history of ionic current-pumping to the onset of bursting.   The assumption here is that the subsequent spikes in a burst are a "ringing" effect due to the close coupling of each neuron to itself, rather than a direct effect of the input signals.   Therefore, if they were used as reference points for stacking the input PRF signals, they would only contribute noise and computational burden.

1.3   SOFTWARE DEVELOPMENT OBJECTIVES

The structural design of the signal processing software that is presented later in section 2 is guided by the following considerations.   First is the need to make efficient use of the processing hardware in the MASSCOMP 5700 so that the least amount of available data from the MMEP apparatus is lost.   Preliminary timing calculations have shown that so long as the preprocessing tasks (tasks 1 and 2 from page 3) must be handled by the MASSCOMP it will not be possible to do any burst detection or network analysis in real time, and probably only the top 6 to 10 channels in the priority list can be reduced to spike data.   The remaining analysis tasks will have to be done off-line in non-realtime using previously saved spike data.

However, the next consideration is that hardware development is under way for the offloading of the preprocessing from the MASSCOMP to an array of TMS 32020 processors.   Therefore, the processing software needs to be flexible in its scheduling of processing tasks, according to whether its data is coming directly from the MMEP in raw form, indirectly from the MMEP through the 32020 boards as spike data (but still in real time), or directly from disk or tape storage as spike data or burst data on demand.

The fact that the software is being developed in response to experimental necessities requires careful attention to structured programming and modular design.   Our initial expectations of the signal processing routines that will be effectual for the desired analyses will have to be modified with experience.   The ability to

hang new and different subroutines into the scheduler without
incurring timing or synchronization problems that propagate
unchecked through the rest of the processes must be built in from
the start.

Because of the limitations on the size and the intensity of
the programming effort, it is recognized that it will not be
feasible to attempt to implement a large and complex software
system. On the other hand, it is highly likely that a low level
of programming effort will be applied to the project over a number
of years. It is wise, then, to take a lesson from this
investigator's past: In 1980 I was assigned to restructure a
system analysis program that was written to predict the
performance of a large solar photovoltaic energy system. That
program was begun small and grew as the system developed. It was
in the form of a single FORTRAN main program without a single
subroutine call aside from intrinsic functions! It was several
thousand lines of incomprehensible rat's nest. I am not
suggesting that anyone on this project would be quite that crude.
Rather, I am emphasizing that it is all right to design a modular,
structured, and comprehensive signal processing program that is
perhaps overwhelming in its scope, but that is at least not likely
to have to be razed several years down the road. With that
thought in mind, we proceed to the structural design of the
algorithm.

2.0   STRUCTURAL DESIGN

The structural design of the processing software is specified
by the HIPO ("Heirarchy, Input-Process-Output") charts which are
included in Appendix C to this report. The following paragraphs
are intended to elaborate on those charts to assist the
programming team in their implementation.

2.1 TOP LEVEL HIPO DESCRIPTION

The top level HIPO chart contains five primary processes.
The first process controls the initialization of the program and
its parameters to reflect the experimental configuration and the
proper disposition of output data (filenames, display devices,
etc.). The user also specifies his processing priorities to
override defaults that will be accepted by the scheduler.

The second process accepts data from the selected source
devices or data files, and performs "learn mode" operations on it
in non-real time. These operations provide the preliminary
pattern recognition functions to establish thresholds, identify
clusters and cluster centers, and tailor a processing sequence to
each channel for the benefit of the master scheduler.

The third process accepts data from the designated source and
applies the appropriate data compression subroutines in accordance
with the processing requirements and timing limitations. This

10

process does not analyze the compressed data (except insofar as
some form of analysis is inherent in the compression), but instead
provides various levels of compressed data to facilitate the
analysis routines in the fourth process.   (The functions to be
performed are described in paragraph 1.2 and in Appendices A and
B.)

    The fourth process applies certain statistical and analytical
functions in accordance with user specifications and timing
limitations.   This process obtains data from the various levels of
compression for graphical representation, listings, etc.; it
computes histograms, correlograms, stacks; and it provides
transfer of compressed data and processing results to appropriate
output files/devices.   This process provides the primary user
interaction with the ongoing experiment.

    The fifth process terminates the experiment.   It is
responsible for purging buffers, closing files, appending user-
supplied text to archive files, and the like.

3.0   REFERENCES

1.   Gerstein, George L., Donald H. Perkel, and Judith E.
      Dayhoff, "Cooperative Firing Activity in Simultaneously
      Recorded Populations of Neurons: Detection and Measurement",
      J. Neuroscience, Vol 5, No. 4, pp. 881-889, April, 1985.

2.   Hestenes, David, "How The Brain Works: the next great
      scientific revolution", unpublished manuscript, Department of
      Physics, Arizona State University.

3.   MacGregor,   ., and Lewis,   ., Neuronal Modeling, Plenum
      Press, 1977.

4.   Okada, Masahiko, and Maruyama, Naoshige, "Software system for
      real-time discrimination of multi-unit nerve impulses",
      Computer Programs in Biomedicine, Vol 14, pp 157-165, 1981.

5.   Rohling, Hermann, "Radar CFAR Thresholding in Clutter and
      Multiple Target Situations", IEEE Trans. on Aerospace and
      Electronic systems, Vol AES-19, No. 4, July 1983.

6.   Schmidt, Edward M., "Computer separation of multi-unit
      neuroelectric data: a review", J. Neuroscience Methods, Vol
      12, pp. 95-111, 1984.

7.   Tou, J.T., and Gonzalez, R.C., Pattern Recognition
      Principles, Addison-Wesley, Reading, Mass., 1974.

APPENDIX   A

CLUSTERING ALGORITHMS

A.1   A ONE-PASS ALGORITHM

The following algorithm works well when the data are grouped
into clusters whose diameter is less than the distance to their
nearest neighboring cluster.   Significant violation of this
condition will make the algorithm highly sensitive to the arbi-
trary choices imposed in the ordering of the data.

In the following description, the data points   X, Y, etc.,
may be taken to be scalar voltage samples, in the case where we
are looking for clusters in the amplitude data; or they may be
taken to be the vectors of dimension   N   consisting of the first
. N   samples including and following a threshold crossing, in the
case where we are looking for clusters in the pulse waveform
types.   In either case, the notation $|X-Y|$ means the Euclidean
distance between the points, whether they are in one dimension or
in   N   dimensions.

Let $(X1,X2,...,Xn)$ be a sequence of   n   data points (or
vectors), and define Z_i to be the i-th cluster center.   The
algorithm finds the set $\{Z_i\}$ of cluster centers.   First, it is
required to obtain a cluster diameter,   D,   and we assume here that
we can obtain it by experimentation (to see which values produce
the most reasonable clusterings) or by a-priori knowledge of the
variance in amplitudes from a single emitter.

Having established   D   we then define $Z1 = X1$.   Then, for
$i = 2$ to n , compute the distance from X_i to each of the cluster
centers, Z_j.   If the distance is greater than   D   for each   j,
then add X_i to the set of cluster centers.   Otherwise, assign
X_i to the first cluster for which $|X_i-Z_j| < D$.

If the clusters are sufficiently well-defined that this is a
reasonable algorithm to use, then we recommend a post-clustering
step to redefine the cluster centers (which will be used later for
the realtime amplitude discrimination) to be the centroids of the
individual clusters, found by vector or scalar averaging.

A.2   The MAXIMIN ALGORITHM

The Maximin clustering algorithm, like the one described
above, is a heuristic procedure but in this case multiple passes
through the data are required.   The difference is that the Maximin
looks for clusters that are farthest apart first, and instead of
having to know an explicit feature of the clusters in advance

(i.e., the diameter, D), we have to experiment with a more
nebulous parameter, F, which is a number between 0 and 1.
Start with F = 0.5 for now.

     As with the one-pass algorithm, we begin by selecting the
first datum (scalar or vector) X1 to be the first cluster center,
Z1. For the second cluster center, Z2, we find the data point
that is farthest from Z1. (If the distance between Z1 and Z2 is
sufficiently small, we can declare that there is only one cluster
and quit.) Let A1 be the distance !Z2-Z1!.

     Suppose we now have a set {Z1,...,Zm} of cluster centers, a
number A(m-1) which is the average of the previous *maximum*
distances, and let {Y1,...,Yn} be the set of data points that have
NOT been assigned to clusters yet. For each j = 1...m, compute
the distances Dij = !Zj-Yi!, i = 1...n , and save the MINIMUM of
these, say, Dj'. Then find the MAXIMUM of the {Dj' : j=1,...,m}.
Call it D'. if D' is greater than F*A(m-1) then declare the
sample corresponding to D' to be a new cluster center, Z(m+1),
and compute the new average maximum distance with D' included.
Otherwise, terminate the algorithm.

APPENDIX   B

PEAK-FINDING ALGORITHMS

B.1   THE GRADIENT-SIGN-CHANGE ALGORITHM

It is well-known from elementary calculus that an extremum of a differentiable function occurs wherever the sign of the derivative (gradient) changes from positive to negative, or vice versa.   This fact has been used by numerous authors (cf. [4]), and it provides a fast and easy computational method whenever the signal exhibits well-separated peaks that are well above the noise.   Those conditions seem to apply in the present situation.

The algorithm is applied whenever the thresholding detector has declared that the signal has crossed the threshold, and it continues until an end condition is satisfied.   Let $X1$ be the datum whose absolute value has just exceeded the threshold, and let n be the least number of samples that are ever needed at the present sample rate to cover any spike waveform in the data.   For each $j = 1 \ldots n$ we check that the sign of $(Xj-X(j-1))$ is different from the sign of $(X(j+1)-Xj)$.   Zero is included as a possible third "sign".   If the sign has changed, then an extremum of height (or depth, if negative) Xj is declared to have occurred at the index (time) j, and the current index is given an increment (in ADDITION to the normal loop increment) so that in case $(X(j+1)-Xj)$ was zero the next point will not also be declared as a peak.

The algorithm continues until a predetermined number of peaks have been found, or until the   n-th   datum following the threshold crossing has been tested, whichever occurs first.   Processing the n-th datum without finding a peak must be reported as an error.

The manner of detecting that the two differences have changed sign depends on the number of CPU clock cycles that are required for a multiply instruction.   If the product of the adjacent differences is less than or equal to zero, then the sign has changed.   However, if a multiply is too costly, then the logical decisions can be streamlined by keeping track of whether the threshold crossing was a negative crossing or a positive crossing and using one of two sequences of logical tests, one being optimized for ascending data, the other for descending data, with a switch being made after each peak is found.

The reason that one might want to find more than one peak in the waveform is that it provides an additional parameter for amplitude discrimination that might be used successfully to avoid having to classify the peak with a Euclidean metric in 20 to 50 dimensions.

APPENDIX   C

"HIFO" CHARTS

INPUT		OUTPUT
	====> 0.0 MMEP LAB SIGNAL PROCESSING AND ANALYSIS PROGRAM ====>	PIPED-UP SYSTEM
USER DATA FROM FILES ====> OR KEYBOARD	1.0 PROGRAM INITIALIZATION/ USER INTERUPT	====>
MMEP RAW DATA ====>	2.0 DATA ACQUISITION/ DISPLAY	====> DATA TO DRIVE SIMULATED OSC. PARAMETERS OF RAW DATA RAW DATA HISTOGRAMS RAW DATA FILE RACETRACK BUFFER LEARN OUTPUT
RACETRACK BUFFER ====> ELECTRODE PRIORITIES ELECTRODE CHARACTERISTICS (TIME TAGED) OR SPIKE DATA /OFFLINE\ OR BURST DATA \PROCESS/	3.0 DATA PROCESSING	====> SPIKE CHARACTERIZATION: SOURCE, AMPLITUDE, TIME BURST CHARACTERIZATION: SOURCE, START, STOP, FREQ, F(AMP) NETWORK CHARACTERIZATION
SPIKE DATA ====> BURST DATA NETWORK DATA	4.0 PROCESSED DATA STATISTICS	====> PARAMETERS OF SPIKES PARAMETERS OF BURSTS PARAMETERS OF NETWORK PROCESSED DATA HISTOGRAMS
USER REQUESTS ====>	5.0 PROGRAM WRAP UP	====> DATA FILES LISTINGS PLOTS DISPLAYS

INPUT   =====> 1.0 PROGRAM INITIALIZATION/   OUTPUT
                                                USER INTERUPT

USER DATA FROM FILES =====> 1.1 COLLECT NON-DEFAULT   =====> PIPED-UP SYSTEM
OR KEYBOARD   PROCESSING PARAMETERS AND
                                     OPTIONS FROM THE USER

KEYBOARD COMMANDS   =====> 1.2 USER INTERUPT   =====> RE-FIPED SYSTEM

SIGNAL PROCESSING SYSTEM

INPUT		1.1 COLLECT NON-DEFAULT PROCESSING PARAMETERS AND OPTIONS FROM THE USER	====>	OUTPUT
KEYBOARD INPUTS	====>	1.1.1 SET EXPERIMENTAL PARAMETERS	====>	PROCESSING & CONTROL PARAMETER ARRAYS/FILES
PROCESSING & CONTROL PARAMETER ARRAYS/FILES		1.1.2 INITIATE DATA COLLECTION		
		1.1.3 MULTICHANNEL OSCILLOSCOPE CHANNEL/SCAN SELECTION		
		1.1.4 SET CHANNEL AND PROCESSING PRIORITIES		
		1.1.5 SPECIFY ARCHIVING REQUIREMENTS		
		1.1.6 INITIATE ARCHIVING		
		1.1.7 TERMINATE ARCHIVING		
		1.1.8 SELECT PLOT AND LISTING OPTIONS		
		1.1.9 SET DEBUG FLAGS		
		1.1.10 SELECT OFFLINE PROCESSING (POSTPROCESSING OPTION)		
		1.1.11 SELECT DISPLAY WINDOWS/ ASSIGNMENTS		

INPUT		OUTPUT
KEYBOARD INPUTS	====>	PROCESSING & CONTROL PARAMETER ARRAYS/FILES
PROCESSING & CONTROL PARAMETER ARRAYS	====>	

====>	1.2	USER INTERUPT

====>

1.2.1 MULTICHANNEL OSCILLOSCOPE CHANNEL/SCAN SELECTION

1.2.2 SET CHANNEL AND PROCESSING PRIORITIES

1.2.3 SPECIFY ARCHIVING REQUIREMENTS

1.2.4 INITIATE ARCHIVING

1.2.5 TERMINATE ARCHIVING

1.2.6 TERMINATE DATA COLLECTION

1.2.7 SELECT PLOT AND LISTING OPTIONS

1.2.8 SET DEBUG FLAGS

1.2.9 SELECT OFFLINE (POSTPROCESSING OPTION)

1.2.10 SELECT DISPLAY WINDOWS/ ASSIGNMENTS

SIGNAL PROCESSING SYSTEM

INPUT	====>	2.0 DATA ACQUISITION/DISPLAY	====>	OUTPUT
MMEP RAW DATA	====>	2.1 GET MMEP DATA INTO MAIN MEMORY	====>	RACETRACK BUFFER OF MMEP DATA
USER SELECTED CHANNELS RACETRACK BUFFER	====>	2.2 RAW DATA GRAPHICS AND STATISTICS PARAMETRIC STATISTICS -- THOSE NEEDED FOR SIMULATED OSCILLOSCOPE -- RANGE OF SAMPLES (MAX/MIN) -- MEANS, MEDIANS, MODES, S.D., etc	====>	DATA NEEDED TO DRIVE SIMULATED OSC. DATA FOR HISTOGRAM PLOTS DATA FOR LISTINGS RAW DATA FILE STORED SPIKE TEMPLATES
RACETRACK BUFFER OR STORED LEARN OUTPUT	====>	2.3 LEARN	====>	LEARN OUTPUT: ELECTRODE ACTIVITY, COMPLEXITY, AND CHARACTERIZATION AT TIME = t (WHENEVER WE RELEARN THEN THAT CHANNELS CHARACTERIZATION, ACTIVITY, COMPLEXITY IS AS OF A NEW TIME = t')

SIGNAL PROCESSING SYSTEM

INPUT		OUTPUT
	====> 2.2 RAW DATA GRAPHICS AND STATISTICS	
RACETRACK BUFFER	====> 2.2.1 PASS DATA THROUGH	====> RAW DATA FILE
PROCESSING AND CONTROL PARAMETERS ARRAYS		
RACETRACK BUFFER	====> 2.2.2 SIMULATED OSCILLOSCOPE	====> RAW DATA TRACE FOR SELECTED CHANNELS STORED SPIKE TEMPLATES
PROCESSING AND CONTROL PARAMETERS ARRAYS		
RACETRACK BUFFER	====> 2.2.3 RAW DATA STATISTICAL PROCESSING	====> PARAMETRIC STATISTICS RAW DATA HISTOGRAMS
PROCESSING AND CONTROL PARAMETERS ARRAYS		

SIGNAL PROCESSING SYSTEM

INPUT		2.2.2 SIMULATED OSCILLOSCOPE ====>	OUTPUT
RACETRACK BUFFER	====>	OBJECTIVE: TO DUPLICATE AND ====>	RAW DATA TRACE FOR SELECTED CHANNELS
PROCESSING AND CONTROL PARAMETERS ARRAYS		EXTEND THE FUNCTIONS OF THE ANALOG OSCILLOSCOPE, THUS PROVIDING A MEANS OF VERIFYING DATA AND ALGORITHM FIDELITY, SELECTING CHANNELS TO PROCESS, AND EXTRACTING TEMPLATES	STORED SPIKE TEMPLATES
		FUNCTIONS:	
		1. MANUAL SELECTION OF CHANNELS OR AUTOMATIC SCAN FOR ACTIVE CHANNELS	
		2. MANUAL/AUTOMATIC ADJUSTMENT OF VERTICAL SCALE, HORIZONTAL SWEEP RATE (DECIMATION RATIO)	
		3. SELECT SWEEP OR SCROLL MODE	
		4. FREEZE DISPLAY WITHOUT INTERRUPTING DATA COLLECTION, PROCESSING, ARCHIVING, IF IN PROGRESS	
		5. SELECT & EXTRACT TRACE FOR ASSIGNMENT TO TEMPLATE LIBRARY	
		6. EVENT TRIGGERING	

INPUT		3.0 DATA PROCESSING		OUTPUT
	====>		====>	
TIME TAG OF DATA BEING WRITTEN TO BUFFER	====>	3.1 EVALUATION OF LAG VARIABLE	====>	SPEED-UP/SLOW-DOWN PROCESSIONG FLAG
TIME TAG OF DATA BEING READ FROM BUFFER				
USER CHANNEL SELECTION FOR PROCESSING	====>	3.2 SELECTION OF PROCESSING SUBROUTINES AND CHANNELS TO BE PROCESSED	====>	CHANNEL AND PROCESSING PRIORITIES
SPEED-UP/SLOW-DOWN PROCESSING FLAG				
RACETRACK BUFFER	====>	3.3 MASTER PROCESSING LOOP	====>	SPIKE AND/OR BURST AND/OR NETWORK DATA
LEARN OUTPUT				
RACETRACK BUFFER	====>	3.4 ADAPTATION/RELEARNING ON ON SELECTED CHANNEL(S) SEE 2.3	====>	LEARN OUTPUT
USER CHANNEL SELECTION FOR RELEARNING				

SIGNAL PROCESSING SYSTEM

INPUT	====>	3.2 SELECTION OF PROCESSING SUBROUTINES AND CHANNELS TO BE PROCESSED	====>	OUTPUT
SPEED-UP/SLOW-DOWN FLAG	====>	3.2.1 RACE CONDITION A (PROCESSING TOO FAST, WAITING ON NEW DATA)	====>	PROCESSING SEQUENCE FOR CHANNELS AND PROCESSING DEPTH
USER CHANNEL SELECTION FOR PROCESSING/RELEARNING:		-- SELECT CHANNELS AND/OR PROCESSES FOR INCLUSION INTO MASTER LOOP AND INCLUDE THEM		
CHANNEL PRIORITY FROM 2.3:				
USER SELECTION OF PROCESSING DEPTH (SPIKE DETECT, BURST DETECT, NETWORK DETECT, STATISTICAL PARAMETERS)				
SPEED-UP/SLOW-DOWN FLAG	====>	3.2.2 RACE CONDITION B (PROCESSING TOO SLOW, NEW DATA WRITING OVER CURRENT DATA)	====>	PROCESSING SEQUENCE FOR CHANNELS AND PROCESSING DEPTH
USER CHANNEL SELECTION FOR PROCESSING/RELEARNING:		-- SELECT CHANNELS AND/OR PROCESSES FOR EXCLUSION FROM MASTER LOOP AND EXCLUDE THEM. SKIP DATA IF NECESSARY		
CHANNEL PRIORITY FROM 2.3:				
USER SELECTION OF PROCESSING DEPTH (SPIKE DETECT, BURST DETECT, NETWORK DETECT, STATISTICAL PARAMETERS)				

INPUT		3.3 MASTER PROCESSING LOOP		OUTPUT
RACETRACK BUFFER LEARN OUTPUT	===>	3.3.1 PREPROCESSING SEQUENCE	===>	SPIKE DATA CLASSIFICATION & CHARACTERIZATION
SPIKE DATA ARRAY (FROM REALTIME OR FROM ARCHIVE)	===>	3.3.2 BURST PROCESSING SEQUENCE	===>	BURST CHARACTERIZATION
BURST DATA ARRAY (FROM REALTIME OR FROM ARCHIVE)	===>	3.3.3 NETWORK PROCESSING SEQUENCE	===>	NETWORK CHARACTERIZATION

CHANNEL SOURCE	AMP.	DETECTION SAMPLING INTERVALS
1, 1	XXX	100, . . .
1, 2		165, . . .
1, 3		172, . . .
2, 1		153, . . .
1, 1		100, . . .

SIGNAL PROCESSING SYSTEM

INPUT		OUTPUT
	====> 3.3.1 PREPROCESSING SEQUENCE ====>	
RACETRACK BUFFER	====> 3.3.1.1 THRESHOLD CROSSING PROCESSING ====>	TIMES WHERE THRESHOLDS ARE CROSSED
APPROXIMATE SPIKE TIMES FROM THRESHOLD CROSSINGS RACETRACK BUFFER LEARN OUTPUT	3.3.1.2 PREPROCESSING SUBROUTINES ====>	SPIKE AMPLITUDE, TIME & CLASSIFICATION AS TO SOURCE

INPUT	====>	3.3.1.2 PROCESSING SUBROUTINES	====>	OUTPUT
THRESHOLD CROSSING TIMES RACETRACK BUFFER LEARN OUTPUT	====>	3.3.1.1 PEAK LOCATION (ONE SOURCE ON CHANNEL) FIND THE HIGHEST (IN ABSOLUTE VALUE) PEAK IN THE 20 MILLISECONDS, AROUND THE THRESHOLD CROSSING. AND / OR	====>	TIME OF ARRIVAL AND AMPLITUDE OF LARGEST PEAK IN ABSOLUTE VALUE, NUMBER OF PEAKS & POSSIBLY THE SOURCE
AMPLITUDE, TIME OF ARRIVAL, NUMBER OF PEAKS LEARN OUTPUT	====>	3.3.1.2 AMPLITUDE DISCRIMINATION IDENTIFY SOURCE BY AMPLITUDE-TIME RECTANGLE OR BY NUMBER OF PEAKS IN PULSE, OR BOTH OR	====>	SPIKE AMPLITUDE, TIME OF ARRIVAL AND SOURCE
AMPLITUDE AND TIME OF ARRIVAL VECTOR OF DATA AROUND THE TIME OF ARRIVAL FOR PEAK LEARN OUTPUT	====>	3.3.1.3 TEMPLATE MATCHING COPY DATA FROM RACETRACK BUFFER TO VECTOR PROCESSOR AMPLITUDE AND TIME OF ARRIVAL OF PEAK TO VECTOR PROCESSOR FOR REGISTRATION OF TEMPLATES INITIATE VECTOR PROCESSOR	====>	SPIKE AMPLITUDE, TIME OF ARRIVAL AND SOURCE